UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Elettronica

TESI DI LAUREA

Piattaforma di accelerazione hardware Xilinx Versal

Laureando: Alessandro Pallaro
Relatore: Daniele Vogrig

ANNO ACCADEMICO 2021/2022

Indice

Elenco delle figure

1. Introduzione

La necessità del mercato globale di comunicare e elaborare informazioni il più velocemente possibile sta spingendo le industrie a sviluppare dispositivi in grado di offrire prestazioni elevate, pur mantenendo versatilità, semplicità e un basso impatto energetico.
Le soluzioni adottate dal 1990 in poi si basavano sul miglioramento dell'efficienza e delle performance dei processori sviluppando diverse architetture quali CISC[1], RISC[2] e strutture multi-core, permettendo di ottenere ottimi miglioramenti nel corso del tempo. Negli ultimi anni però tale approccio non sta producendo i risultati sperati, quindi si è reso necessario sperimentare nuove tipologie di processore per gestire la grande mole di dati da elaborare.
Una soluzione è stata l'impiego di DSP e GPU, ovvero architetture di processori in grado di elaborare tipi di dati molto complessi. Solo particolari applicazioni però riescono a trarne vantaggi e lo sviluppo di queste architetture presenta ostacoli simili a quelli delle CPU. L'incremento delle prestazioni infatti si traduce spesso in aumento dei consumi e dei costi di produzione.
Contemporaneamente anche le FPGA sono state prese in considerazione. Grazie alla loro versatilità hanno avuto largo successo nei vari settori del mercato ma il miglioramento delle performance richiede molto sviluppo a livello hardware.
Arrivando così ai tempi recenti dove ha preso piede lo studio e l'implementazione di tecniche di machine learning e intelligenza artificiale. Sono cosi nate le CPU ibride dotate di core dedicati all'intelligenza artificiale con lo scopo di migliorare le prestazioni e l'efficienza in applicazioni come guida autonoma, riconoscimento del viso, sorveglianza e simili.
L'implementazione di queste novità da parte delle grandi aziende però viene spesso limitato dalla complessità di installazione. Per implementare sistemi con intelligenza artificiale è necessario hardware dedicato e varie modifiche ai sistemi già presenti.
La nuova piattaforma di accelerazione hardware, sviluppata da Xilinx, risponde alle necessità del mercato combinando le varie soluzioni in un unico dispositivo ibrido. Il suo vantaggio è l'essere implementabile su qualsiasi sistema tramite una comunissima interfaccia PCIe.
Lo scopo di questo lavoro di tesi è di elencare le funzionalità di questa piattaforma e discuterne potenzialità e vantaggi che può portare in vari settori industriali.

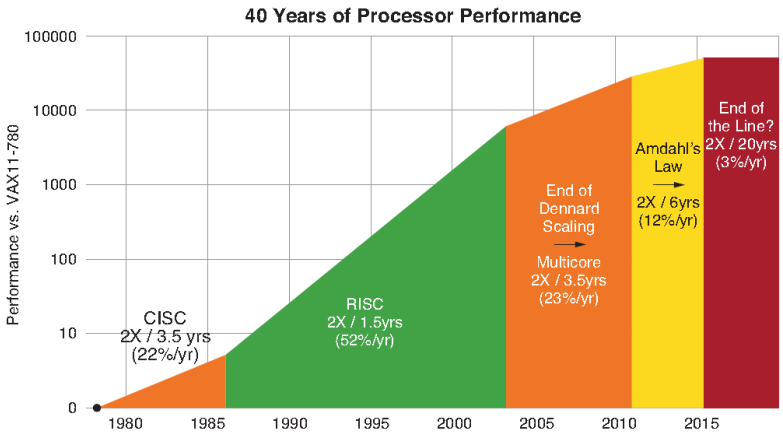

Figura 1.1: Andamento nel tempo delle prestazioni delle CPU
[3, Pag.2]

2. Architettura del chip Xilinx Versal

Xilinx Versal ACAP[1] è una grande famiglia di dispositivi ibridi, divisa in serie a seconda dell'ambito per cui i diversi dispositivi sono stati progettati.

La serie Versal Prime è la base della famiglia Versal. Può essere sfruttata in una grande varietà di applicazioni offrendo modesti incrementi in performance rispetto alle tradizionali CPU. Alcuni ambiti in cui i dispositivi possono essere sfruttati sono i network e Data Center di medie e grandi aziende, in cui possono migliorare le prestazioni delle comunicazioni e dei dispositivi di archiviazione dati. La serie può offrire vantaggi anche nelle aziende di data streaming e nell'ambito dell'aereo-nautica e difesa.

La serie Versal AI Edge garantisce discrete performance nell'ambito dell'intelligenza artificiale. E' progettata per sistemi di controllo in tempo reale che richiedono un interpretazione intelligente dei dati in ingresso, come ad esempio la guida autonoma.

La serie AI Core è focalizzata sull'intelligenza artificiale, consentendo di ottenere incredibili performance nelle applicazioni vettoriali. Il processore AI, grazie all'elevata densità di core, è in grado di elaborare una grande mole di dati ed è adatto a una varietà di applicazioni, come ricerca scientifica, grandi data center, servizi Cloud e streaming video.

Infine Versal Premium presenta le prestazioni più elevate della famiglia Versal. Con una banda complessiva di ben 18Tb/s e molti processori DPS si integra perfettamente in ambiti in cui è richiesto un elevato traffico di dati e elevate performance computazionali come comunicazioni cablate, data center e laboratori con sistemi sofisticati di misura. I processori di questa serie non sono però dotati di un processore dedicato all'intelligenza artificiale, quindi non sono adatti a svolgere compiti di machine learning e elaborazione intelligente di dati.

I vari dispositivi della famiglia Versal sono divisi in serie a seconda dei componenti che integrano e delle performance che essi riescono a offrire. Nonostante vi siano differenze, a volte molto significative, tra una serie e l'altra, è possibile definire un'architettura generale sulla quale tutti i dispositivi si basano. Tale architettura può essere divisa nei seguenti blocchi fondamentali:

- PMC per l'avvio del dispositivo, monitoraggio e servizi del sistema.
- Processing system
- Logica Programmabile
- Processore per l'elaborazione di segnali digitali (DSP)
- Interfaccia PCIe realizzata tramite logica programmabile.
- Network on Chip programmabile
- Controller per accedere alla memoria RAM DDR4
- I/O programmabile ad alta velocità (XPIO)

[1] Adaptive Compute Acceleration Platform

Ai quali si possono aggiungere, a seconda della serie:

- Processore per l'intelligenza artificiale
- Sezione di logica programmabile più estesa
- Un CPM con DMA e memoria Cache, per una comunicazione PCIe più performante
- 4MB di memoria RAM veloce per accelerazione hardware

La presenza di tanti blocchi da origine ad un complesso sistema interconnesso.

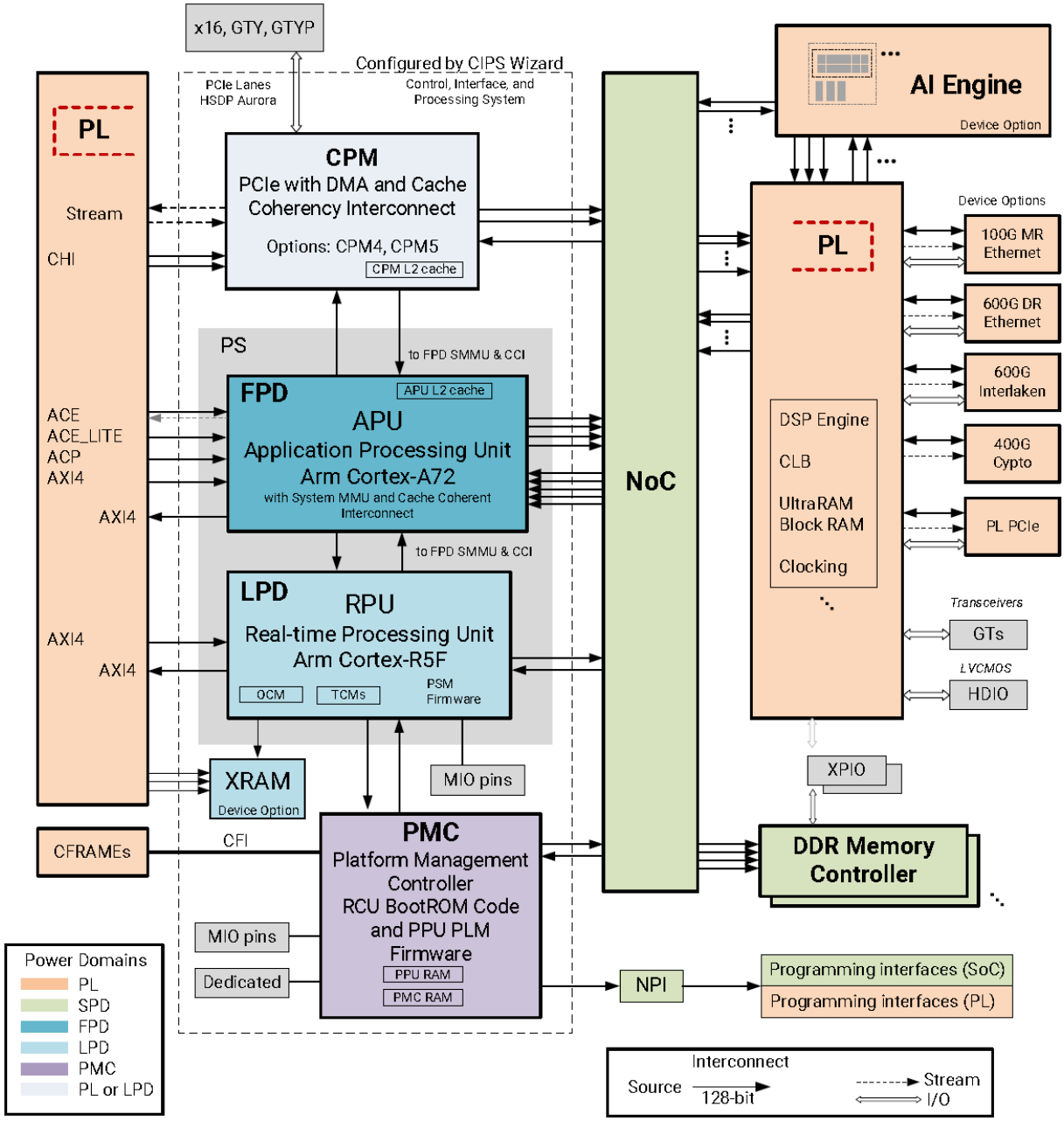

Figura 2.1: Schema a blocchi generale di un dispositivo Versal [4, Pag.46]

Le connessioni tra i vari blocchi non rappresentano però il layout fisico del dispositivo, che è il seguente:

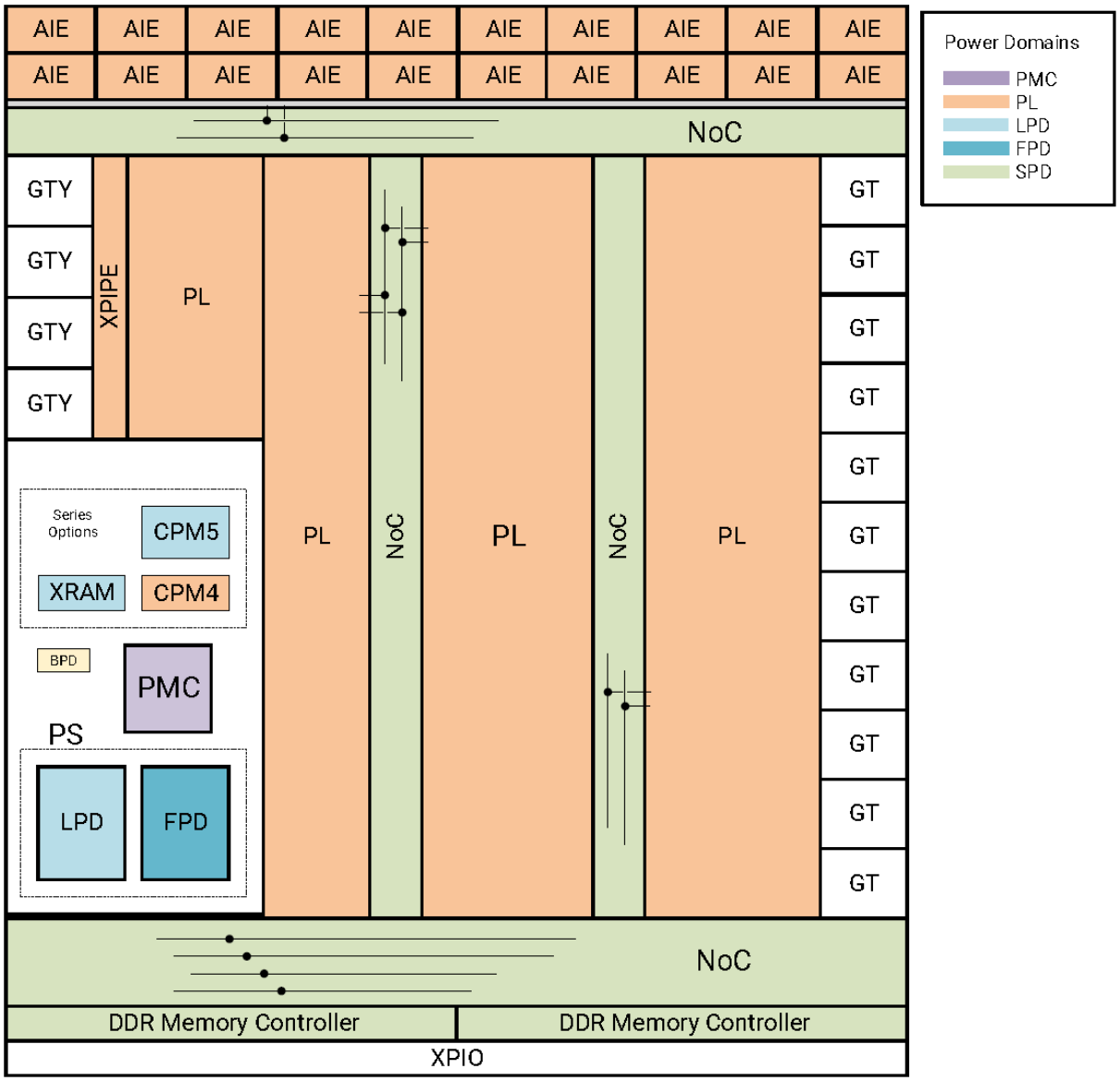

Figura 2.2: Layout effettivo di un dispositivo Versal [4, Pag.54]

Il layout evidenzia come ciascun dispositivo Versal sia complesso e ricco di circuiti integrati con soluzioni programmabili e personalizzabili dall'utente. Elementi come logica programmabile e processing system devono essere configurati via software per lavorare al meglio. Entrambi hanno molti componenti e funzionalità che possono essere attivati o disattivati dinamicamente in modo da poter ottenere una configurazione performante e allo stesso tempo efficiente del dispositivo.
Inoltre il sistema di ciascun dispositivo è monitorato al fine di rilevare e correggere errori, garantendo cosi sicurezza e affidabilità. Questo richiede la presenza di un sistema di controllo (PMC) a bordo del chip, in grado di gestire i vari clock, monitorare lo stato del sistema, il consumo energetico e eventualmente resettare parti del sistema in caso di malfunzionamenti.

3. Platform Management Control

Il primo blocco fondamentale è il Platform Management Control (PMC). Come suggerisce il nome questa sezione del chip è dedicata alla gestione dell'intera piattaforma di accelerazione hardware. Nel dettaglio si occupa di effettuare il boot all'avvio del dispositivo e di configurarlo. Successivamente, durante il normale funzionamento ne monitora i parametri fondamentali, come tensioni e temperature dei processori. Il monitoraggio consente di raccogliere informazioni sullo stato del sistema a cui è possibile avere accesso tramite delle apposite porte di debug.

3.1 Boot e configurazione del dispositivo

L'avvio avviene tramite un processo di boot suddiviso in più fasi, che consente anche di effettuare un avvio normale o un avvio sicuro.

La fase preliminare consiste in un reset generale (memorie volatili e processori) e nella lettura dei parametri scelti dall'utente. Sul dispositivo sono presenti 4 pin che rappresentano un numero binario a 4 bit, il cui valore viene letto all'inizio della procedura di boot e salvato in un registro chiamato Boot Mode Register. I 16 valori rappresentati dai bit indicano la modalità di avvio e la sorgente dei dati di configurazione, specificando anche parametri di comunicazione e il grado di sicurezza desiderato. Per conoscere a cosa corrispondono i 16 valori è necessario consultare i datasheet [5, Pag.63].

Fatto ciò inizia il boot effettivo. All'interno del PMC è presente una BootROM, ovvero una memoria ROM con le istruzioni per il boot al suo interno che vengono eseguite da un processore dedicato, sempre collocato nel PMC. ROM e processore formano la RCU e le prime operazioni che esegue consistono nella lettura del valore salvato nel registro Boot Mode, da cui verrà dedotto il dispositivo primario di avvio.
Gli input da cui è possibile prelevare le informazioni sono 4:

- Porte Quad SPI e Octa SPI, per trasferimenti seriali a quattro o otto canali.
- Slot scheda SD
- Scheda eMMC per le Multimedia Card (Scheda SD con adattatore)

Dopo opportune verifiche sui dati contenuti, il boot prosegue secondo i parametri dettati dal dispositivo. Nel caso in cui non venga trovato nessun dispositivo esterno o i dati non siano corretti il boot avviene comunque secondo la procedura standard. La procedura di boot infatti è completamente scritta nella BootROM, solo alcuni parametri possono essere modificati.
Impostati i parametri del boot, viene avviato un secondo processore contenuto in una sezione del PMC chiamata PPU. Questa sezione è molto semplice e contiene solo un processore e una memoria RAM di appoggio. Il suo scopo è eseguire un programma chiamato Platform Loader and Manager che viene preso dallo stesso dispositivo del boot.

La terza fase del boot è quindi la configurazione della piattaforma da parte del PLM.
Il dispositivo di memoria oltre ai parametri di boot contiene il PLM e i file che specificano:

- Configurazione del PMC stesso
- Configurazione del Processing system
- Software del Manager del Processing system (PSM)
- Configurazione della logica programmabile (File CFI dedicato)

- Configurazione del processore per l'intelligenza artificiale
- Inizializzazione del NoC e NPI (Interfaccia di programmazione del NoC)
- Configurazione di NoC, NPI, memorie DDR, ricetrasmettitori GT, I/O, periodi di clock e altro.

Inoltre, un altro aspetto fondamentale di cui si occupa la PPU è il caricamento dei dati nelle varie memorie dei processori principali, presenti in processing system e AI Engine, sfruttando un file ELF dedicato. E' quindi di cruciale importanza che i file contenuti nel dispositivo siano corretti per non effettuare configurazioni errate del dispositivo.

Ora il boot è terminato e il primo processore si spegne in attesa del prossimo riavvio per ripetere il boot. Invece il secondo processore, nella PPU, esegue la seconda parte del PLM che consiste nel monitoraggio della piattaforma.
Sempre nel PLM è presente una parte di codice, una funzione, denominata Dynamic Function Exchange (DFX) che può essere richiamata dal Processing system e eseguita dalla PPU. Essa consente di cambiare dinamicamente la configurazione della logica programmabile e avviene in modo simile a quello che accade durante l'avvio. Occorre prestare attenzione a specificare un secondo dispositivo di boot, in quanto nel primo è presente la configurazione già usata. Essendo il dispositivo già avviato e funzionante, vengono rese disponibili anche le interfacce PCIe e Ethernet come possibili sorgenti da cui prelevare i dati della configurazione.
La possibilità di effettuare modifiche in tempo reale sulla sezione di logica programmabile rende molto versatile il dispositivo, consentendogli di adattarsi a requisiti di sistema variabili.

Boot Sicuro e non sicuro

Durante la prima fase del boot, i 4 pin indicano anche se l'avvio deve essere sicuro o meno.
Un boot normale avviene caricando consecutivamente i vari file e le varie configurazioni senza eseguire alcun tipo di controllo sui dati contenuti nel dispositivo. Si tratta quindi un operazione molto veloce, però si corrono rischi nel caso in cui non si sia certi della correttezza e della provenienza del codice.

Un boot sicuro invece garantisce autenticità e integrità del firmware che verrà caricato nel dispositivo. Durante un avvio sicuro la RCU si occupa di autenticare o decriptare il software PLM. Una volta che il PLM è ritenuto autentico viene eseguito dalla RCU e contiene le istruzioni per caricare i file di configurazione e di autenticarli a sua volta.
Le fasi di autenticazione possono essere più o meno serrate e ricorrono a vari algoritmi di decifratura, controllo per integrità e parità dei bit. Il prezzo da pagare sono i tempi di avvio più lenti e la necessità di avere un software originale con configurazioni prive di errori.
Il motivo dell'esistenza del boot sicuro è dare la garanzia che il software caricato sia compatibile e impedire che del codice errato possa arrecare danni alla piattaforma.

3.2 Monitoraggio del sistema

Il PMC è in grado di monitorare le tensioni di alimentazione e le temperature di PS, AI Engine e logica programmabile. Questi controlli sono fondamentali per migliorare la sicurezza generale e l'affidabilità della piattaforma. Il sistema di monitoraggio si basa su un convertitore analogico digitale a 10bit a cui è possibile accedere direttamente dal PS tramite varie interfacce e dalla logica programmabile tramite il Network on Chip.
Se le temperature dei processori diventano troppo elevate, a causa di un carico eccessivo per un tempo prolungato o di una inefficiente soluzione per il raffreddamento, il dispositivo potrebbe rompersi istantaneamente o deteriorarsi molto velocemente. Per ovviare a questa possibilità il PMC può ridurre la tensione di alimentazione dei processori per limitarne la potenza e contrastarne l'aumento di temperatura. Limitando la potenza le performance caleranno ma il dispositivo non subisce danni.
Inoltre se la temperatura è accessibile al di fuori della piattaforma Versal è possibile collegare un sistema di raffreddamento esterno che contrasta le temperature elevate solo quando necessario.

3.3 Interfacce per i dispositivi di memoria esterni

Per l'avvio del dispositivo è necessaria la comunicazione con dispositivi esterni.
Il PMC integra un controller per consentire la lettura dei dati dalle schede SD e eMMC [6] che supporta varie velocità di clock per la lettura, da lenta a ultra-alta (UHS), e integra un DMAC per migliorare le prestazioni.

La comunicazione può avvenire anche tramite interfaccia SPI [7].
Nel PMC è presente un secondo controller dedicato, in grado di gestire entrambe le interfacce quad-SPI e octa-SPI in quanto la differenza tra le due è solo il numero di canali su cui avviene il trasferimento dati. In particolare con 8 linee l'interfaccia octa-SPI riesce a raggiungere la velocità di 400MB/s.

4. Processing system

Il processing system è il centro computazionale del dispositivo ed è composto di due regioni principali: il full power domain e il low power domain.

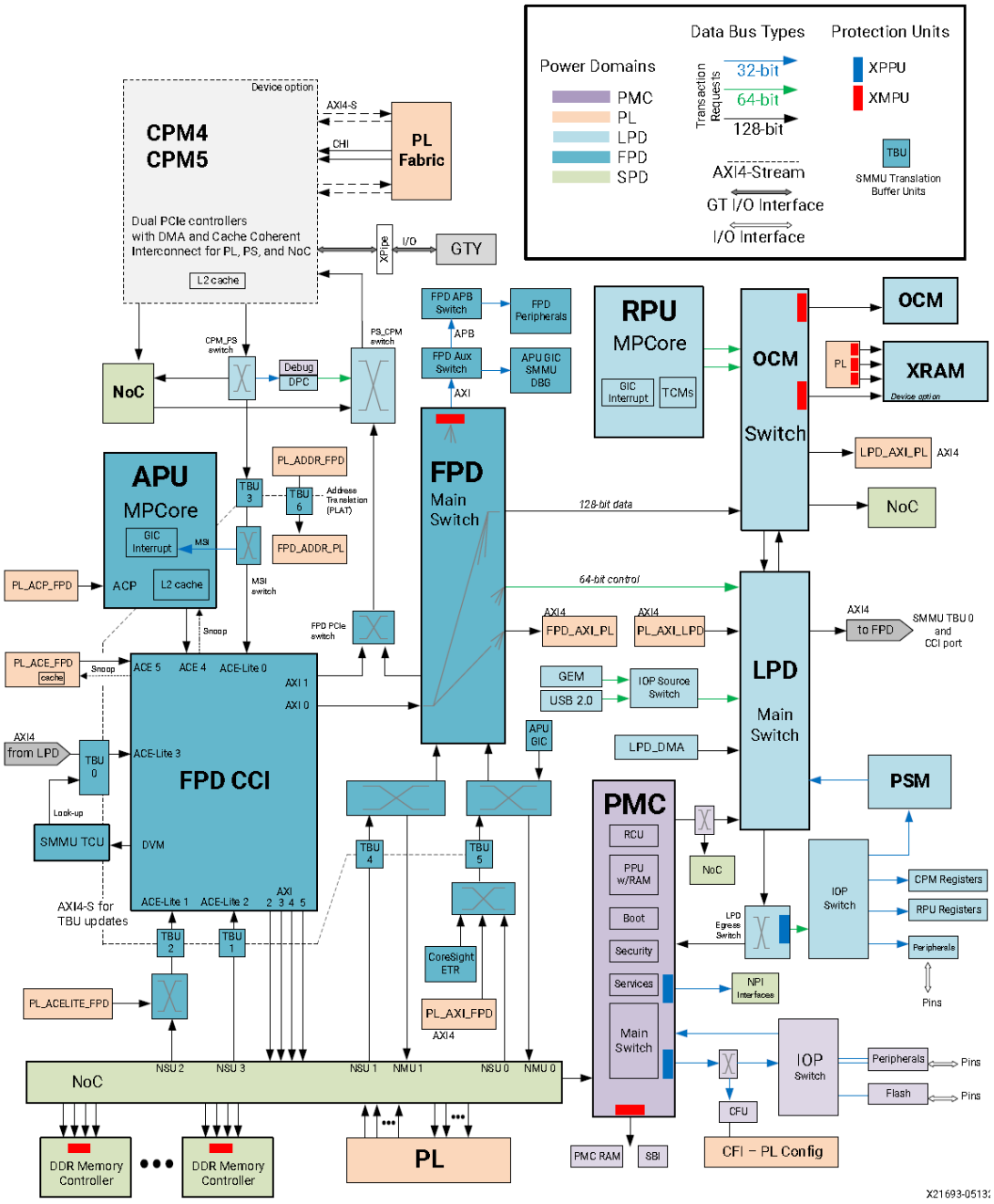

Figura 4.1: Layout generale di Xilinx Versal ACAP
[4, Pag.46]

Il processing system è gestito dal PSM, un vero e proprio processore situato nel Low Power Domain che controlla le varie isole del PS. Le istruzioni per la gestione del PS sono contenute in un firmware che viene scaricato dal PLM e copiato sulla RAM del PSM. Il processore del PSM è un MicroBlaze[8] a 32bit con TMR[9], progettato da Xilinx. La struttura TMR prevede tre sistemi computazionali semplici (core e memoria cache) che eseguono un determinato processo in modo sincrono. L'output prodotto dai tre sistemi viene controllato e se i core producono lo stesso risultato allora viene ritenuto corretto, ottenendo cosi un efficiente metodo di correzione degli errori. Lo scopo del PSM è gestire dinamicamente i registri di controllo del PS, tramite i quali gestisce l'alimentazione delle varie isole. La precisione e il determinismo forniti dal MicroBlaze sono fondamentali in quanto un errore nell'esecuzione può causare sovratensioni sui processori, distruggendoli.

Le isole sono delle zone interne al processing system più piccole dei vari domini, il cui controllo è lasciato al PSM.
In particolare le isole sono:

- Due isole dedicate ai due core dell'APU, una per core.
- Memoria cache dell'APU
- Processore dell'RPU
- 16 isole dedicate ai 4MB di memoria XRAM

Tramite il PSM è quindi possibile gestire le varie modalità di funzionamento del dispositivo risparmiando energia quando non è necessario l'utilizzo di APU e RPU, oppure abilitare a piena potenza i vari processori quando necessario.

4.1 Full Power Domain

All'interno del processing system è situato il Full Power Domain.
E' il centro computazionale principale del dispositivo, sede dell'Application Processing Unit (APU) su cui è possibile eseguire applicazioni di carattere generale, inclusi sistemi operativi.
Oltre all'APU, che sarà analizzata nel dettaglio successivamente, in questo dominio sono presenti un gestore per gli interrupt e un'unità per la gestione della memoria.

Il gestore degli interrupt (GIC-500 [10]) consente la comunicazione tra APU e periferiche. Il suo scopo è gestire le richieste di interrupt generate dalle varie periferiche. Quando viene generato un interrupt, ad esempio la pressione di un tasto sulla tastiera, il processore deve prendersene carico. Se sono presenti più periferiche è possibile vengano generati più interrupt contemporaneamente andando a creare confusione nel processore. Il gestore serve a arbitrare le richieste di comunicazione, consentendo di definire priorità e logiche di gestione.

L'unità per la gestione della memoria di sistema è chiamata SMMU [11]. Il suo compito è suddividere lo spazio degli indirizzi virtuali (l'intervallo di indirizzi accessibili dal processore) in pagine di memoria dimensione 2^N. Gli N bit meno significativi dell'indirizzo (l'offset all'interno della pagina) rimangono invariati, mentre i bit restanti rappresentano il numero virtuale della pagina. La SMMU dispone di diverse unità TBU[1], in cui sono presenti diverse tabelle. Le tabelle consentono di ottenere il numero fisico della pagina corrispondente a quello virtuale, che, combinato con l'offset della pagina, forma l'indirizzo fisico completo.

[1] Translation Buffer Unit

Controllando e aggiornando dinamicamente le look-up table caricate nelle sette unità di buffer delle traduzioni (TBU), l'SMMU si occupa di:

- Traduzione degli indirizzi da virtuali a fisici
- Consentire un trasferimento corretto dei dati tra memorie e processore
- Salvaguardia dei dati nella memoria, è importante che non vadano persi nella virtualizzazione.

Cache Coherent Interconnect

Infine è presente un blocco chiamato Cache Coherent Interconnect [12] per la coerenza della memoria [13].
L'incoerenza della memoria cache è un problema che si presenta in sistemi con struttura multi-core. Generalmente ciascun core si appoggia alla sua memoria cache locale, prelevando e salvando i dati su essa durante l'esecuzione di un programma. Se il programma volesse sfruttare entrambi i core, i dati che verrebbero modificati in una delle memorie dovrebbero essere resi disponibili e aggiornati anche per l'altro core. Fare in modo che la cache sia coerente significa quindi fare in modo che dati presenti in memorie separate possano essere condivisi da più core o processori.
Dunque, nei dispositivi Versal il ruolo del CCI è assicurarsi che tutti i processori e i relativi core abbiano una visione comune dello spazio di memoria.

Nello schema a blocchi in Figura 4.2 si nota che il CCI è connesso all'APU e al NoC, questo perché la memoria RAM principale è accessibile tramite NoC (come si vedrà nel capitolo dedicato). Il CCI è anche connesso al Low Power Domain che contiene la RPU, consentendo cosi la comunicazione diretta tra le cache dei due domini del PS. Infine vi è una connessione alla logica programmabile (PL) perché è fondamentale il trasferimento dei dati per realizzare funzioni specifiche personalizzate.
La comunicazione è resa possibile tramite il protocollo AMBA AXI4 e ACE [14]. I dati trasmessi secondo questi protocolli specificano anche gli indirizzi di memoria di partenza e destinazione. Tramite le TBU il CCI e l'SMMU virtualizzano le memorie, creando un unico spazio di memoria condiviso da tutti i processori.

In seguito è riportato uno schema a blocchi del FPD. Si può notare come altri processori e bus possono interagire con la cache L2 dell'APU, andando a formare un sistema interconnesso.

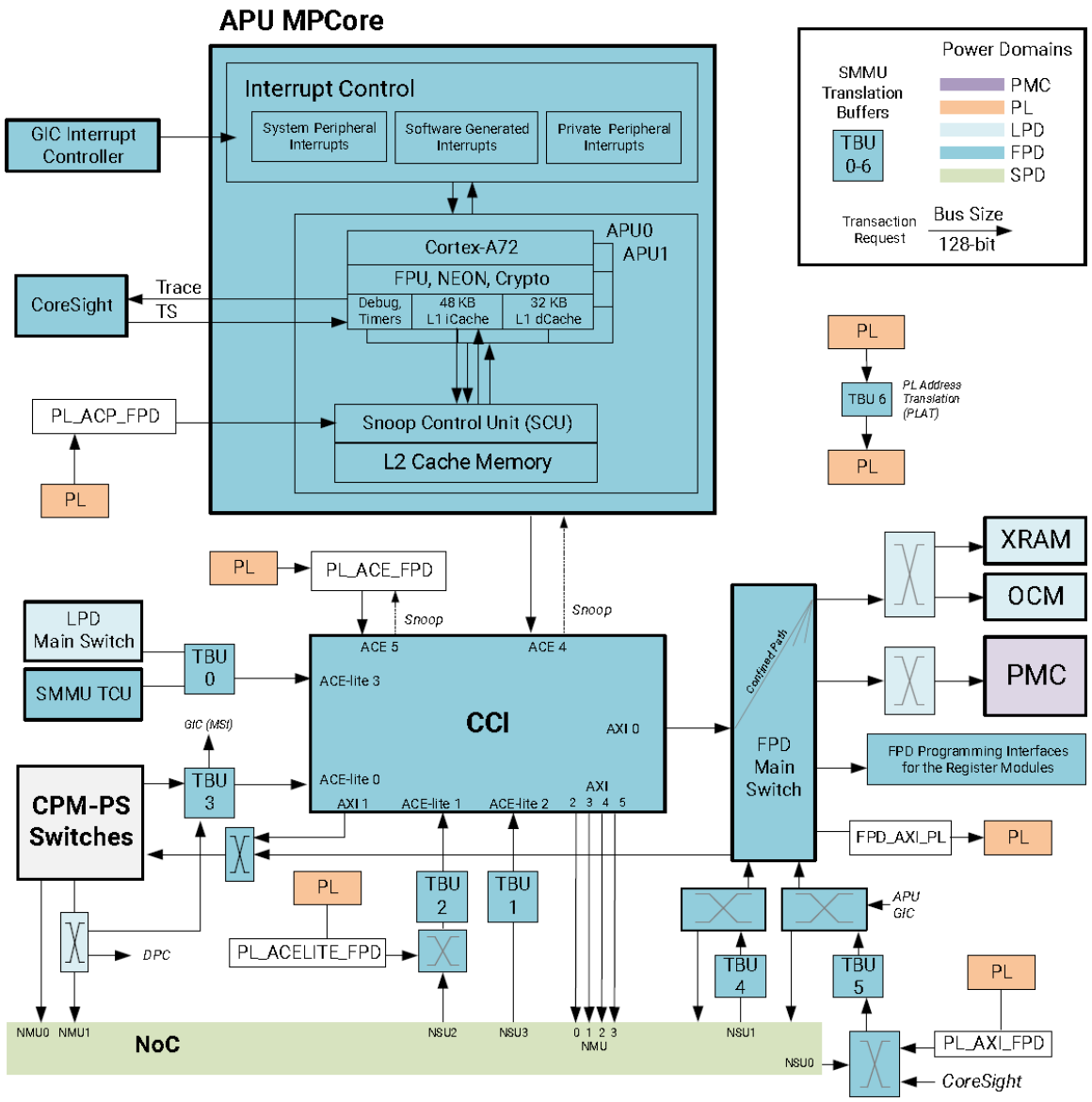

Figura 4.2: Schema a blocchi del FPD
[4, Pag.58]

Application Processing Unit

L'APU è l'unità riservata all'esecuzione di applicazioni di carattere generale. E' progettata per l'elaborazione di dati generici e l'esecuzione di algoritmi e programmi con un elevato carico sulla CPU. Può essere sfruttata per l'esecuzione di sistemi operativi, interfacce di comunicazione e elaborazioni complesse.

Il cuore dell'APU è un processore ARM dual-core a 64bit: il Cortex-A72 con architettura v8-A. E' un processore avanzato, ricco di funzionalità e compatibile con molti software e compilatori grazie alla popolarità dei processori ARM.
Tra le funzionalità dell'architettura v8-A vi è ad esempio la virtualizzazione, che consente a più software di venire eseguiti dallo stesso core del processore o di dividere il carico di un programma su entrambi i core del processore, quindi multi-threading e multi-tasking.
La virtualizzazione può offrire diversi vantaggi a seconda dell'ambito di utilizzo del chip. In generale può fare in modo che il processore sia sfruttato al pieno delle sue capacità, aumentando le performance generali e riducendo i tempi di esecuzione di programmi che lavorano in parallelo. In casi più rari invece può essere usato come un modo per partizionare i vari stack dei programmi per isolarli o creare ridondanze.

Come memorie di appoggio per il processore sono predenti due memorie cache. Entrambi i core del processore hanno a disposizione 48KB di memoria cache L1 per le istruzioni e 32KB di memoria cache L1 per i dati, entrambe completamente associative e con protezione ECC. Poi è presente 1MB di cache associativa a 4 vie, di tipo L2 e con protezione ECC. Anch'essa può essere raggiunta dalla sezione di logica programmabile e dal CPM tramite le varie interfacce AXI/ACE.
Non è presente una memoria RAM nell'APU perché il processore sfrutta la memoria principale del dispositivo, accessibile tramite il DDRMC del NoC.

La CPU integra anche tre pipeline:

- Una pipeline per l'unità per l'aritmetica a virgola mobile a singola e doppia precisione su dati di 32 e 64 bit.
- Una pipeline a lunghezza variabile super-scalare fino a 15 stadi. Capace di predizione dinamica della fase di branch, con buffer per il branch target e cronologia di branch.
- Una pipeline con decodifica neon per le istruzioni SIMD, ovvero istruzioni che consentono operazioni su più dati alla volta. Grazie alla presenza di istruzioni SIMD è possibile realizzare un Cryptography Engine che può essere usato per accelerare l'esecuzione di algoritmi AES [15], SHA e SHA2-256 [16].

4.2 Low Power Domain

Il secondo dominio del processing system è il Low Power Domain.
In questo dominio sono presenti:

- Real-time Processing Unit
- PSM per la gestione dell'alimentazione dei processori, visto nel capitolo 3
- Memoria on Chip (OCM)
- OCM Switch per la gestione della memoria RAM
- Blocchi per la gestione delle periferiche
- Memoria XRAM

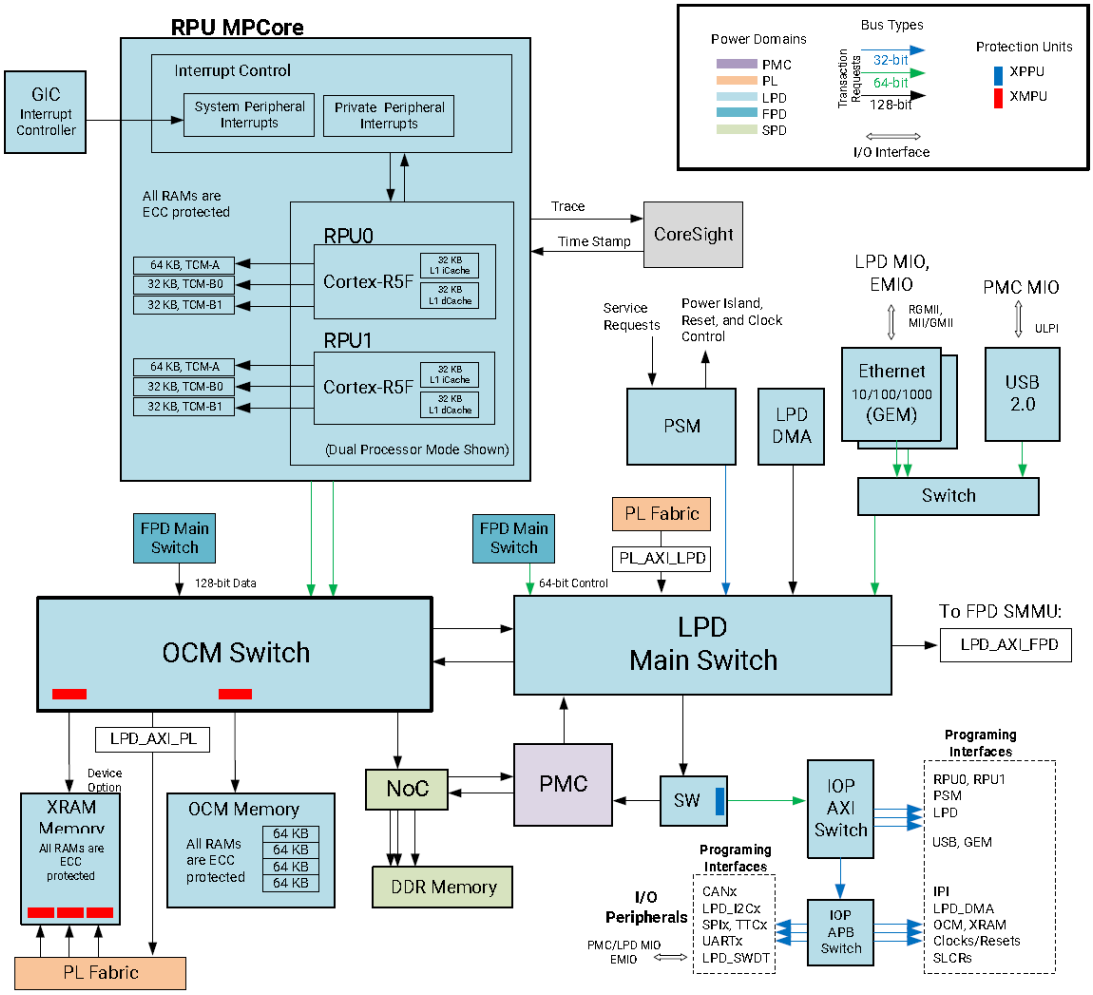

Figura 4.3: Schema a blocchi del LPD
[4, Pag.61]

Real-time Processing Unit

L'unità per l'elaborazione di processi in tempo reale è formata da una CPU ARM Cortex-R5F.
E' un processore dual-core con frequenza ridotta rispetto all'APU ma in grado di eseguire efficientemente operazioni in virgola mobile a singola o doppia precisione.
La sua peculiarità è l'essere configurabile per lavorare in modalità multi-core quando sono richieste le massime performance, oppure poter operare in modalità lock-step quando sono richiesti determinismo e sicurezza d'esecuzione.

La modalità lock-step viene usata in applicazioni in cui è di fondamentale importanza non commettere errori. I due core del Cortex R5F lavorano in parallelo ma con qualche ciclo di offset tra di loro. Alla fine di ogni ciclo l'output del core in ritardo viene confrontato con l'output ottenuto dall'altro core e se ci sono differenze l'errore viene segnalato e può essere gestito dal sistema.
Inoltre, il processore si appoggia su memorie RAM di tipo TCM [17] con ECC. Ciò significa sia che i tempi di accesso alla memoria sono determinabili a priori sia che le memorie sono in grado di rilevare e correggere errori sul singolo bit. Gli errori di cui si parla non sono causati dal processore o da altri blocchi del sistema ma sono errori noti come bit-flip [18].
Il termine TCM invece indica che le memorie sono molto vicine ai core del processore rendendo possibile una connessione a bassa latenza che consente accessi rapidi completabili in un singolo ciclo di clock.

Per realizzare la modalità lock step è necessaria un'architettura apposita che includa un sistema di confronto degli output e dei meccanismi di sincronizzazione tra i due core.

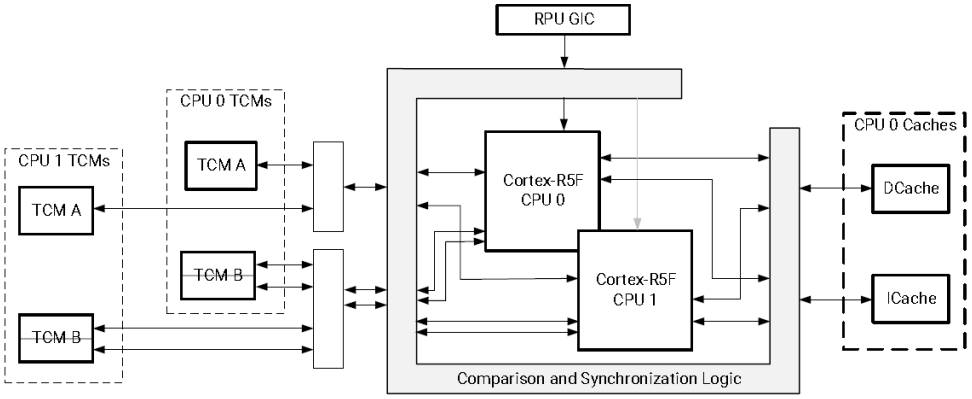

Figura 4.4: Architettura Lock-Step
[4, Pag.217]

La combinazione della lock-step mode con le memorie TCM fa si che il processore possa essere utilizzato nell'analisi di sistemi in tempo reale, in cui è necessario prevedere i tempi di elaborazione dei segnali per riuscire a rispondere velocemente.
L'RPU è dotata di due blocchi TCM, a loro volta divisi in banchi protetti, che offrono ai core del processore una memoria deterministica a bassa latenza. Questo però si traduce in una bassa densità di memoria. In modalità performance, anche chiamata dual-core, ogni core ha a disposizione 128KB. In modalità lock step invece la memoria viene condivisa, arrivando a 256KB.

La bassa densità di memoria delle TCM non pone un problema per questo sistema poiché il suo scopo si basa sull'elaborazione dei segnali provenienti dalle periferiche. I trasferimenti principali vengono gestiti da un DMA e i dati vengono salvati in memoria. In generale la dimensione di essi è ridotta e una volta elaborati possono essere eliminati o salvati in memorie più grandi. In alcuni casi invece la comunicazione avviene direttamente tra RPU e periferiche, di conseguenza è necessario un gestore degli interrupt, il GIC-390 [19] che ha scopo analogo al GIC-500 dell'APU. Le memorie TCM servono solo inoltre per accedere rapidamente ad eventuali parametri e dati di appoggio come valori di riferimento o valori limite.

L'RPU è in grado di sfruttare le memorie in modo diverso a seconda della modalità in cui lavora.:

Configurazione	**Descrizione**	**Core Attivi**
Split	Modalità ad elevate prestazioni, in cui i due core lavorano in modo indipendente, con memorie TCM separate.	Entrambi
Split con un solo core	Uno dei due core può essere tenuto in "reset", consentendo all'altro di accedere a 256KB di memoria.	Solo uno
Lock Step	"Modalità sicura" in cui l'output viene controllato e corretto	Entrambi ma le performance generali sono dimezzate
Sleep	RPU spenta	Nessuno

Il Cortex R5F non dispone solo delle memorie TCM ma è direttamente collegato a delle memorie cache, la cui latenza di accesso è trascurabile per la maggior parte delle applicazioni in tempo reale.
Entrambi i core della RPU hanno a disposizione 32KB + 32KB di memoria L1 per istruzioni e dati. Le memorie sono collegate al sistema CCI del Full Power Domain consentendo cosi la comunicazione tra CPU principale e RPU.

Memorie on Chip e XRAM

Nel low power domain sono presenti anche due memorie aggiuntive: la On Chip Memory e la XRAM. I loro tempi di accesso sono più lenti rispetto alle TMC ma rimangono molto utili perché i dati che contengono possono essere specificati all'avvio del dispositivo direttamente dal PMC.

La memoria OCM, raggiungibile tramite uno switch, mette a disposizione della RPU 256KB di RAM.
Il compito dell'OCM Switch è quello di gestire gli accessi a questa memoria da parte della RPU. Il collegamento attraversa uno switch perché oltre alla RPU vi possono accedere anche logica programmabile, APU e AI Engine tramite NoC. Lo switch indirizza i vari accessi, le richieste provenienti dalla RPU però vengono trattate con la massima priorità quindi non vengono rallentate. Successivamente vengono eseguite le richieste provenienti dall'APU e poi con priorità minori altri eventuali blocchi.

La memoria XRAM è una memoria disponibile in alcuni dispositivi della serie AI Core. Offre 4MB di memoria divisi in 4 banchi con 4 porte per la comunicazione dei dati, consentendo cosi lettura e scrittura simultanei. Le comunicazioni sono anche protette da un unità apposita di Xilinx (XMPU).
Oltre a fornire spazio di archiviazione, lo scopo principale di questa memoria è mettere in comunicazione la RPU con la sezione di logica programmabile, infatti una porta della XRAM può dialogare con le memorie RAM on chip o con l'RPU direttamente, mentre le altre tre porte dialogano con la logica programmabile. La logica programmabile può cosi velocizzare l'esecuzione di funzioni complesse che richiederebbero molto tempo alla RPU, Per questo motivo la memoria viene anche chiamata RAM per Accelerazione Hardware.

La porta da cui accede il LPD ha un'interfaccia AXI4 a 128 bit, mentre le altre tre porte possono essere configurate per dati di lunghezza variabile di 32, 64, 128 o 256 bit. Ciascuna di queste tre porte hanno la loro unità di protezione per facilitare l'accesso simultaneo sia in lettura che scrittura. L'XMPU impedisce che i dati vengano corrotti o letti in modo errato prevenendo scritture e letture su dati in fase di aggiornamento.

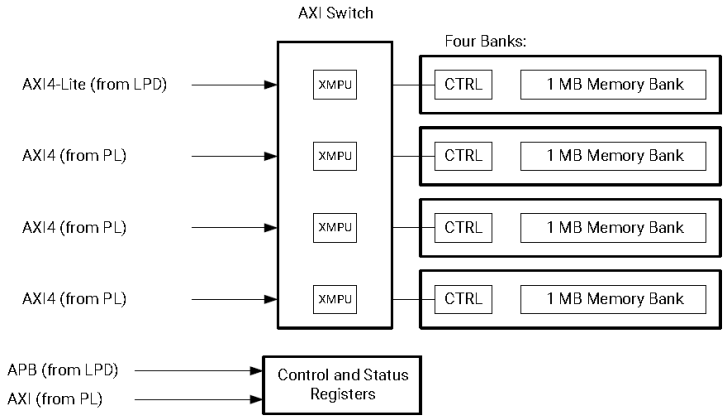

Figura 4.5: Schema a blocchi della XRAM
[4, Pag.465]

XRAM e memorie OCM si interfacciano anche al PMC tramite lo switch. Come anticipato, il PMC è in grado di effettuare una configurazione iniziale dei dati quando il dispositivo viene avviato. In seguito la programmazione della memoria può essere gestita dalla logica programmabile. In questo modo però la porta del LPD viene disabilitata e l'XRAM serve solo come memoria aggiuntiva per la logica programmabile.

5. DMA

All'interno di sistemi complessi con più processori, memorie e tante periferiche, è molto frequente che vi siano richieste di accesso ai dati archiviati nelle varie zone del dispositivo. Per poter gestire lo spostamento di dati tra sorgente e destinazione è necessario un bus su cui viaggiano i dati e un master. Un master è un processore a cui è consentito accedere alla memoria e ha il compito di dirigere gli spostamenti di dati.
In sistemi comuni, in cui si dispone di una CPU molto veloce e poche periferiche, è accettabile che l'unico master sia la CPU stessa essendo già in grado di accedere sia ai dati presenti in memoria sia alle informazioni provenienti dalle periferiche. Questo metodo di trasmissione dati era molto comune nei computer e nelle console di qualche anno fa ed è chiamato Programmable Input Output (PIO). L'APU infatti sfrutta il sistema PIO in quanto le periferiche sono limitate e la CPU ha frequenza di clock elevata. Il problema di questo approccio è che il processore impiega molti cicli di clock per trasferire dati e mentre è impegnato nel trasferimento non può effettuare altre operazioni. Nel caso della RPU, il cui scopo è effettuare comunicazioni continue con le periferiche, il trasferimento dei dati andrebbe a impiegare gran parte delle operazioni svolte dal processore, rallentando l'elaborazione e riducendo drasticamente le performance.
Per questo all'interno del Low Power Domain, cosi come in altre zone del dispositivo, sono presenti delle unità per l'accesso diretto alla memoria (DMA).

Il DMA è un meccanismo che permette l'accesso diretto alla memoria interna per il trasferimento di dati, in lettura e/o scrittura, senza coinvolgere il processore principale.
In un sistema DMA sono presenti almeno due master che possono accedere alla memoria, ovvero un DMA controller e la CPU principale. Il DMA controller svolge gli stessi compiti che svolgeva il processore nel metodo PIO. Si occupa di prelevare i dati dalla sorgente e di portarli a destinazione in maniera trasparente al processore, che può quindi continuare a lavorare normalmente durante il trasferimento dei dati. Il processore rimane comunque un master perché avrà comunque bisogno di accedere a periferiche e memorie quando necessario.
Essendo presenti più master e un unico bus si pone il problema di chi ha l'accesso al bus. Per questo in un sistema DMA è presente anche un bus arbitrator che decide quale master può utilizzare il bus. Usando un sistema di richieste d'accesso e autorizzazione l'arbitrator organizza gli accessi dei vari master in modo che non vi siano conflitti nei trasferimenti di dati.

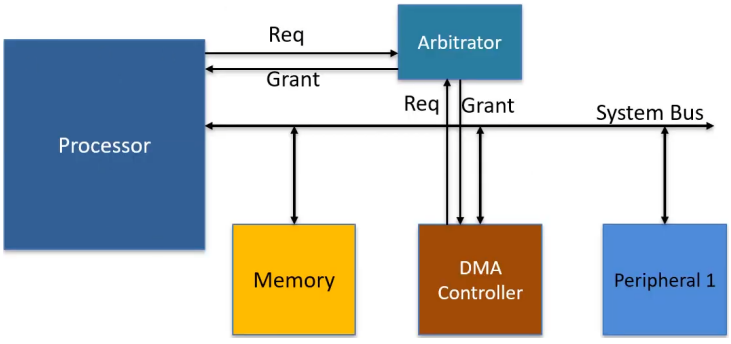

Figura 5.1: Schema di un sistema DMA

Esistono varie logiche con cui l'arbitrator gestisce le molteplici richieste di accesso al bus, ma le più comuni sono:

- Coda semplice
- Coda con priorità
- Scelta casuale
- Assegna ad ogni master una finestra temporale in cui può accedere alla memoria (TDMA)

Un sistema DMA ha il vantaggio di essere realizzato tramite hardware, risultando molto più veloce rispetto al PIO e il processore non viene impegnato durante i trasferimenti.

Lo schema visto in precedenza può essere ulteriormente ottimizzato.
Il trasferimento dei dati avviene copiandoli dalla periferica al buffer interno del DMA controller, per poi essere copiati in memoria o viceversa. Collegando le periferiche direttamente al DMA controller esso si comporta da ponte tra periferiche e memoria. I dati quindi vengono sempre copiati nel buffer interno del DMA, ma mentre vengono copiati possono già essere trasferiti nella memoria dato che il bus è libero. Questo consente di risparmiare molto tempo al costo di un DMA controller più complesso.

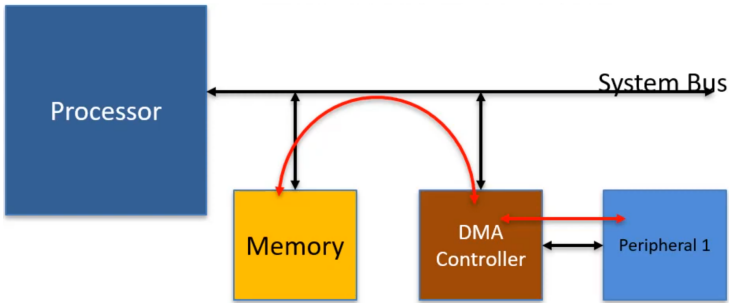

Figura 5.2: Ottimizzazione del DMA

Una volta stabilita la configurazione del sistema DMA e le logiche di gestione dei master, rimane il problema di comandare il DMA controller (DMAC). Esso infatti non può sapere quando deve eseguire un trasferimento di dati, da dove arrivano i dati, dove devono andare e qual'è la dimensione dei dati.
Per questo anche in un sistema DMA è necessario l'intervento del processore, che su necessità delega al DMAC il trasferimento dei dati. I vantaggi del sistema visti in precedenza rimangono, in quanto al processore è necessario comunicare solo pochi parametri fondamentali come indirizzi di memoria e lunghezza dei dati.
Questo richiede che il DMAC abbia dei registri di controllo su cui il processore possa andare a depositare queste informazioni e anche dei registri di stato per poter comunicare con il processore.

DMA controller nel Processing System

Nel low power domain viene reso disponibile un DMA controller collegato a periferiche, APU, RPU, CCI e tramite NoC alla memoria RAM principale. Il suo compito è liberare la RPU dalla gestione delle periferiche ma viene anche sfruttato dalla CCI per gli spostamenti di dati tra le memorie cache. La comunicazione tra periferiche e memoria RAM invece consente di trasferire programmi o grandi quantità di dati.

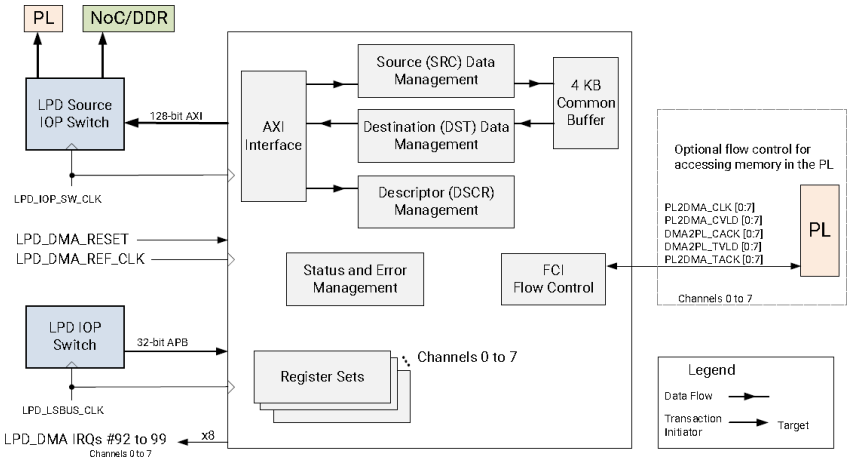

Figura 5.3: Schema del DMAC nel PS [4, Pag.249]

Il DMA controller è dotato di 8 canali di comunicazione indipendenti, controllabili dal processore tramite 8 registri di controllo. Prima di abilitare il trasferimento il processore deve aggiornare sorgente e destinazione su cui il canale opererà, nonché la dimensione dei dati del trasferimento. L'unica eccezione è quando si tenta di accedere alle memorie realizzate con la logica programmabile, in quel caso i segnali di controllo provengono direttamente dalla PL stessa.

Una volta abilitato ogni canale è responsabile per il trasferimento dei dati

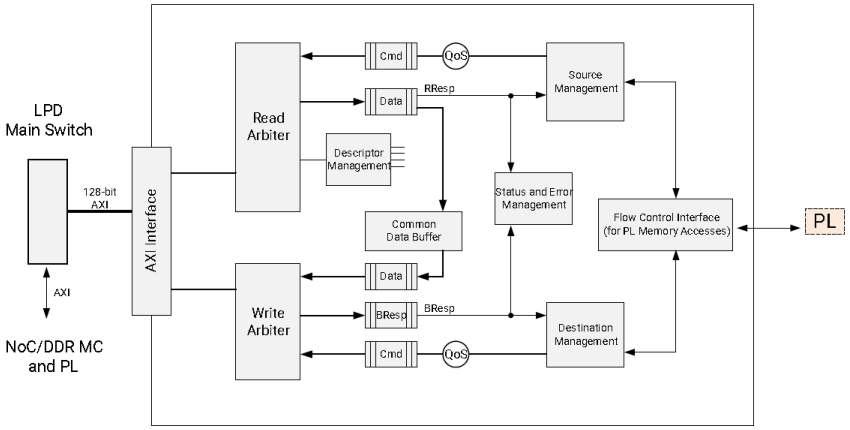

Figura 5.4: Un canale del DMAC [4, Pag.252]

Gli 8 canali dispongono di un buffer interno comune di 4KB su cui appoggiarsi durante il trasferimento dei dati. Il buffer è gestito completamente via hardware, quindi prima di iniziare il trasferimento dei dati non è necessario preoccuparsi di come sono allocati i dati. L'hardware del DMAC assegnerà automaticamente gli indirizzi di accesso al buffer per i canali abilitati, dividendo in modo equo la memoria a disposizione. E' possibile quindi avere lo spazio di memoria diviso in 8 se tutti i canali sono abilitati, mentre un canale avrà a disposizione tutti i 4KB se è l'unico abilitato per il trasferimento.

6. Intelligenza artificiale

Vari settori del mercato, negli ultimi anni, stanno implementando meccanismi di machine learning e intelligenza artificiale per offrire prodotti con funzionalità innovative. E' il caso di videocamere di sorveglianza, smartphone, automobili con guida assistita, servizi di streaming e simili.
Il problema dell'intelligenza artificiale è che l'esecuzione degli algoritmi di apprendimento e decisione non vengono eseguiti efficientemente da una normale CPU. Per questo motivo la serie Versal AI Core e AI Edge sono dotate di un AI Engine Array. Sfruttando l'elevata densità computazionale di questa sezione i dispositivi sono in grado di offrire prestazioni di gran lunga superiori a quelle dii CPU professionali e FPGA in applicazioni che fanno uso di DSP, machine learning e intelligenza artificiale.

All'interno dell'AI Engine Array è possibile individuare due zone.
Una zona è dedicata alla gestione delle comunicazioni con il resto del dispositivo, come memorie, processori e periferiche. Essa prende il nome di AI Engine Array Interface e si compone di celle con ruoli specifici.
La seconda zona è l'AI Engine Array vero e proprio. In questa zona è stata creata una struttura a tabella collegando molteplici celle computazionali (AI Engine Tile) tra di loro. Tale struttura in informatica viene chiamata array bidimensionale e questo spiega l'origine del nome di questa sezione.

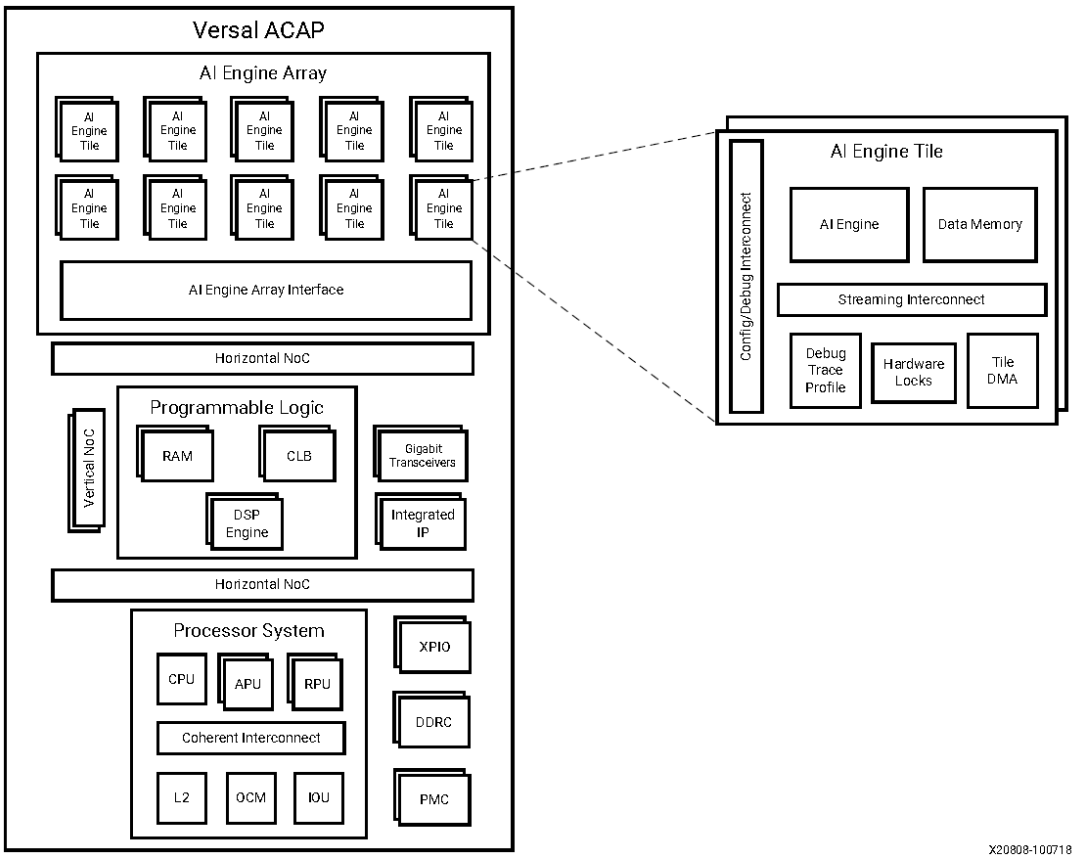

Figura 6.1: AI Engine - Schema a blocchi semplificato
[20, Pag.8]

6.1 AI Engine Array Interface

Il compito dell'interfaccia è consentire a logica programmabile e NoC l'accesso all'array di processori per l'intelligenza artificiale. Anche l'interfaccia, come l'array, si scompone in celle:

- Cella di interfaccia con la logica programmabile
- Cella di interfaccia con il NoC
- Cella di interfaccia per la configurazione, unica per tutto l'array
  Si occupa di:

  Generazione del clock dei processori tramite un PLL [21]

  Gestione di accensione e reset

  Generazione degli interrupt

  Fa da mediante durante il Dynamic Function Exchange invocato dal PMC

  Configurazione dei registri globali dell'array da cui derivano molti parametri di funzionamento.

La cella di configurazione è unica per l'intero array e viene sfruttata solo nella fase di avvio e di DFX. Ciascuna colonna dell'array ha due celle d'interfaccia dedicate per il trasferimento dei dati da elaborare.

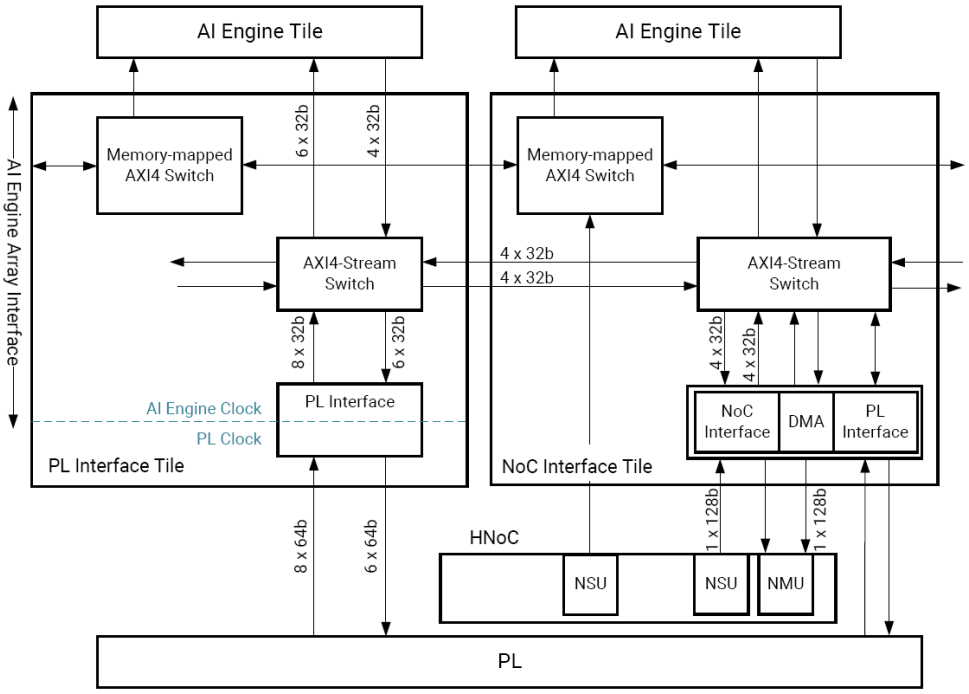

Figura 6.2: Celle di interfaccia con NoC e PS
[20, Pag.37]

Interfaccia con la logica programmabile

Le celle dedicate alla logica programmabile contengono due switch e un modulo di interfaccia (PL Interface).

Quando la logica programmabile ne ha necessità invia all'AI Engine Array dei dati, tipicamente vettori, da elaborare. I clock delle due sezioni però sono diversi, per questo viene interposto un modulo di interfaccia che fa da buffer. Il modulo si comporta come una coda FIFO, ovvero una coda che da priorità ai dati che sono in attesa da più tempo.
La comunicazione è bidirezionale: la logica programmabile ha a disposizione 8 canali a 64 bit per l'invio dei dati mentre l'AI Engine Array può restituire l'output tramite 6 canali a 64 bit.

I dati possono poi essere prelevati dalla coda e inviati alle varie celle dell'array tramite lo switch per l'interfaccia AXI4 Stream [22]. Il ruolo principale dello switch AXI4-Stream è trasportare ad elevate velocità il flusso di dati tra i vari processori dell'array, logica programmabile e NoC.
Ciascuna singola cella di interfaccia mette a disposizione una banda di 32GB/s e 24GB/s (in ingresso e in uscita) per ciascuna colonna dell'array. Generalmente l'array è composto di decine di colonne, ciò significa che le velocità di trasferimento tra logica programmabile e AI Engine sono dell'ordine del TB/s.

Per la configurazione dei processori è invece presente uno switch per l'interfaccia AXI4 mappata. I pacchetti inviati tramite AXI4 hanno 32 bit nei quali è specificato l'indirizzo (colonna e fila) del processore a cui sono indirizzati. Questo switch non serve per il trasferimento di grandi moli di dati ma è utilizzato solo in fase di programmazione.
All'interno di ogni cella dell'array è presente una memoria RAM in cui vengono scritte le istruzioni che ogni AI Engine deve eseguire. Tramite il NoC è possibile trasferire il programma, o la parte di programma che ciascuna cella eseguirà.

Interfaccia con il NoC

Le celle per l'interfaccia con il NoC sono del tutto analoghe alle celle della logica programmabile ma vedono l'aggiunta di un modulo di interfaccia con il NoC e un DMA per alleviare dai processori il compito del trasferimento dati.
Il modulo NoC ha il ruolo di convertire i pacchetti utilizzati dal network in pacchetti da 32 bit compatibili con l'AI Engine. Per l'accesso al NoC la cella d'interfaccia si appoggia alle unità NSU e NMU perché i dati devono venire instradati tra le varie sezioni del dispositivo.

6.2 Cella dell'AI Engine Array

Ogni cella dell'array può essere considerata come un sistema computazionale completo. Dispone infatti di un processore, un modulo di memoria RAM (contenente sia dati che istruzioni) abbinato ad un DMAC e ad un interconnect che consentono la condivisione dei dati con tutte le altre celle dell'array.

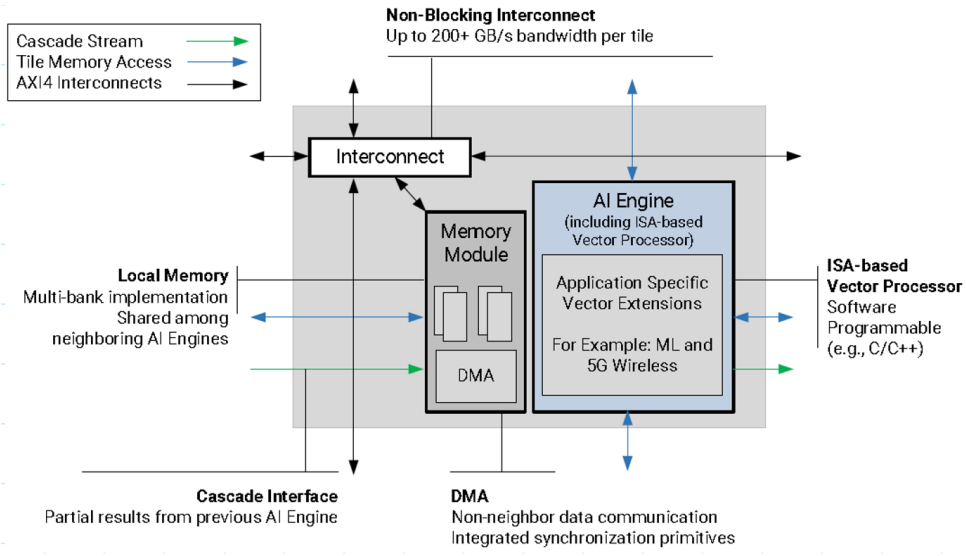

Figura 6.3: Struttura di una cella [20, Pag.13]

Modulo di memoria e interconnect

Prima di descrivere il processore vero e proprio è necessario descrivere il funzionamento del modulo di memoria e dell'interconnect.

L'interconnect è il blocco che gestisce tutte le comunicazioni che avvengono all'esterno della cella. Si occupa di fare da switch per il traffico dell'array e da punto di accesso dei dati per la singola cella.

I pacchetti di configurazioni provenienti dall'interfaccia sfruttano l'AXI4 mappata. L'interfaccia mappata consente a qualsiasi blocco esterno di accedere a registri e memorie della cella, questo significa che qualsiasi master con accesso al NoC può raggiungere la cella al fine di configurarla o modificare il programma che il processore esegue.

Vi sono poi i pacchetti di dati provenienti dallo switch AXI4-Stream. Essendo presente uno switch per ogni colonna l'interconnect ha solo il compito di fare da ponte per consentire al pacchetto di arrivare alla cella di destinazione. Una volta raggiunta la destinazione, l'interconnect di quella cella inserirà i dati nel modulo di memoria.

Il modulo di memoria è collegato direttamente all'interconnect e al processore mettendo a disposizione 32KB di memoria dati divisi in 8 banchi che consentono fino ad 8 accessi simultanei. La gestione degli accessi viene affidata a due DMA, un DMA si occupa del traffico in ingresso, l'altro di quello in uscita. Per evitare problemi di sincronizzazione sono presenti 16 Locks [23] (8 in ingresso per ogni banco e 8 in uscita).

Nella memoria vengono contenuti tutti i dati che devono essere elaborati dall'AI Engine, quindi sia i banchi di memoria che i DMA garantiscono elevate prestazioni. In particolare il modulo presenta anche una connessione diretta all'AI Engine della cella precedente creando un sistema di calcolo a cascata in cui l'output del processore precedente viene direttamente elaborato dalla cella successiva.

AI Engine

Il fulcro di ogni singola cella è l'AI Engine. Si tratta di un processore altamente ottimizzato per l'esecuzione di istruzioni SIMD e VLIW. I due acronimi indicano il fatto che le istruzioni eseguite dal processore sono molto lunghe, ovvero 128 bit rispetto ai 32 o 64 bit tradizionali e che le istruzioni contengono più informazioni. Il vantaggio di avere istruzioni lunghe è la possibilità di effettuare più operazioni contemporaneamente andando a creare un parallelismo di esecuzione.
I 128 bit disponibili vengono scomposti in istruzioni più piccole che possono essere eseguite dal processore in un singolo ciclo di clock. Da una singola istruzione è possibile estrapolare due operazioni scalari, una moltiplicazione vettoriale, due letture e una scrittura di vettori.
Per essere in grado di eseguire le varie operazioni in un unico ciclo il processore dispone di unità dedicate ad ogni compito.
Per l'ambito computazionale ogni AI Engine è dotato di un'unità scalare e un'unità vettoriale.
Il salvataggio e il prelievo dei dati viene eseguito da delle unità di load e store chiamate AGU[1]. Esse si occupano di generare gli indirizzi di memoria che i dati hanno all'interno della memoria dati della cella. Una volta generati gli indirizzi le unità comunicano con i DMA del modulo di memoria per effettuare il trasferimento.
La gestione delle istruzioni viene lasciata ad una unità di fetch e decode dedicata. Essa si occupa di prelevare da una memoria di 16KB interna al processore le varie istruzioni da eseguire. Dopo il prelievo le istruzioni vengono scomposte e inviate alle due unità computazionali.

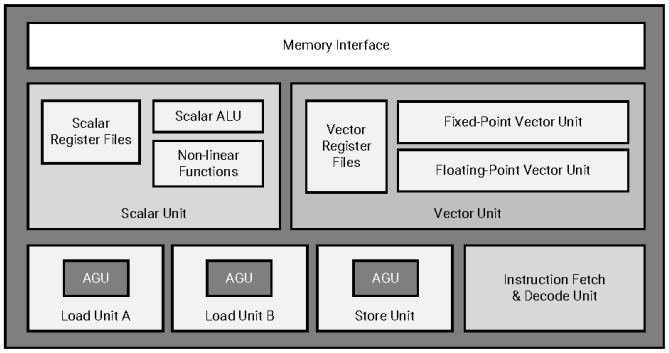

Figura 6.4: Schema del AI Engine [20, Pag.44]

Le unità di calcolo scalare e vettoriale sono a sua volta composte da blocchi logici che si occupano a tutti gli effetti dell'elaborazione.

L'unità scalare si compone di un sommatore RISC a 32bit e un moltiplicatore 32x32bit. Include anche un blocco per le conversione di dati tra rappresentazione in virgola mobile e rappresentazione intera, nonché in grado di eseguire funzioni non lineari. Le funzioni non lineari hanno molti utilizzi nel campo dell'elaborazione dei segnali e quest'unità è in grado di svolgere le più importanti ovvero funzioni trigonometriche, radici e funzione reciproca. Il calcolo di queste funzioni però richiede più di un ciclo di clock.

L'unità di calcolo vettoriale è in grado di gestire vettori in rappresentazione intera o in virgola mobile.
Per la rappresentazione intera l'elaborazione viene effettuata direttamente da tre circuiti logici realizzati tramite hardware. Un circuito MAC consente di sommare e moltiplicare dei vettori, salvandone il risultato in un registro apposito. Il circuito di upshift effettua un semplice shift verso destra. Infine il circuito SRS consente di trasferire un valore intero in un vettore effettuando opportuni arrotondamenti e shift logici.
Se i dati sono rappresentati in virgola mobile, come è comune accada, vengono rese disponibili più funzioni. E' possibile effettuare somme e prodotti scalari tra i vettori, confronti di uguaglianza, maggioranza

[1] Address generation unit

e conversione ad intero. Per la conversione dei dati da rappresentazione intera a floating point è presente un circuito dedicato analogo all'SRS.
L'unità di calcolo vettoriale viene migliorata da un sistema pipeline a 7 stadi. L'esecuzione di un'istruzione normalmente richiede diverse fasi. L'istruzione deve essere prelevata dalla memoria RAM e decodificata, viene poi effettuata la ricerca dei dati per eseguire l'istruzione e immagazzinare i dati. In un sistema senza pipeline le varie unità di fetch, decodifica, store e load vengono lasciate per gran parte del tempo senza nulla da fare aspettando il completamento dell'istruzione. Questo significa che ogni fase richiede un ciclo di clock, quindi ogni istruzione richiede diversi cicli per essere eseguita.
In un sistema con pipeline invece ogni fase dell'esecuzione ha uno stadio indipendente dagli altri. Ogni stadio provvede a eseguire in un singolo ciclo di clock il suo compito, per poi passare direttamente all'istruzione successiva. Questo fa si che si possa considerare che in un sistema con pipeline ogni istruzione richieda un solo ciclo di clock per essere eseguita.
Implementando una pipeline a 7 stadi l'AI Engine può effettuare due operazioni scalari e un calcolo vettoriale ad ogni ciclo.

AI Engine Array

Tramite gli interconnect e gli switch presenti nell'interfaccia è possibile unire le delle singole celle in un array dalle ottime performance computazionali.

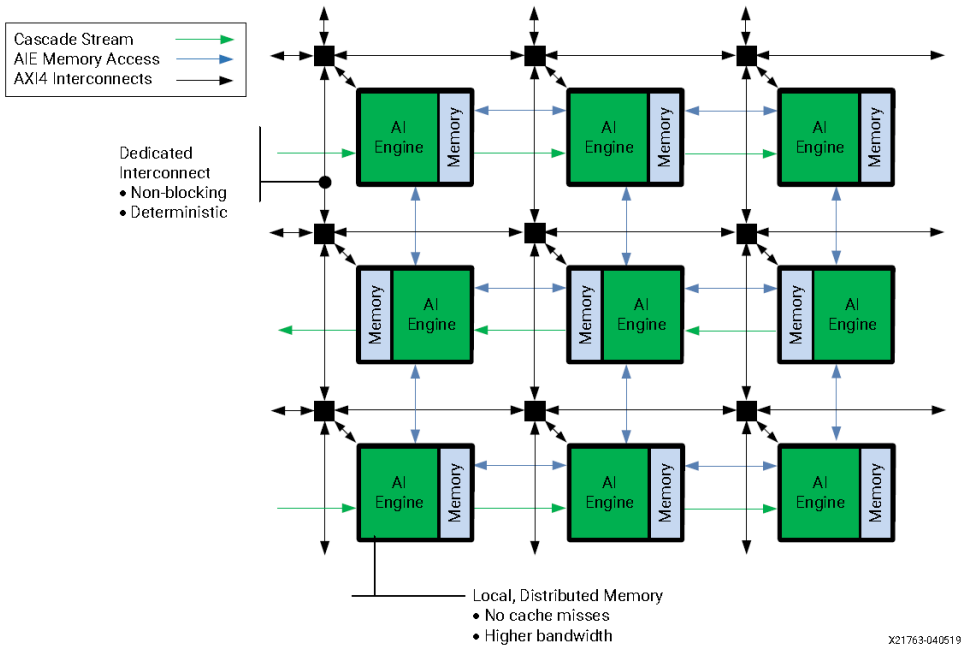

Figura 6.5: Struttura dell'array [20, Pag.15]

Ogni modulo di memoria può comunicare con le 4 celle adiacenti tramite l'interconnect e i vari AI Engine sono connessi direttamente tra di loro per effettuare calcoli in cascata.
Nel complesso si ottiene un processore con una elevatissima densità computazionale. Algoritmi di machine learning e elaborazione di segnali digitali traggono molti vantaggi dall'elevato numero di core del processore.

7. Logica Programmabile

L'alternativa proposta da Xilinx per le soluzioni implementate tramite FPGA è la logica programmabile. Si tratta di una struttura che permette di realizzare una grande varietà di funzioni personalizzate configurando opportunamente delle tabelle di verità. Le funzioni realizzabili sono le più varie, dalla semplice manipolazione dei dati a protocolli di trasferimento, unità computazionali, RAM e altro.

La logica programmabile vuole essere una risorsa di appoggio per i processori principali del dispositivo. Questo rende necessaria la presenza di collegamenti diretti con il processing system, l'AI Engine, il PMC, il NoC e altri blocchi di comunicazione verso l'esterno come XPIO, HDIO e ricetrasmettitori GT.

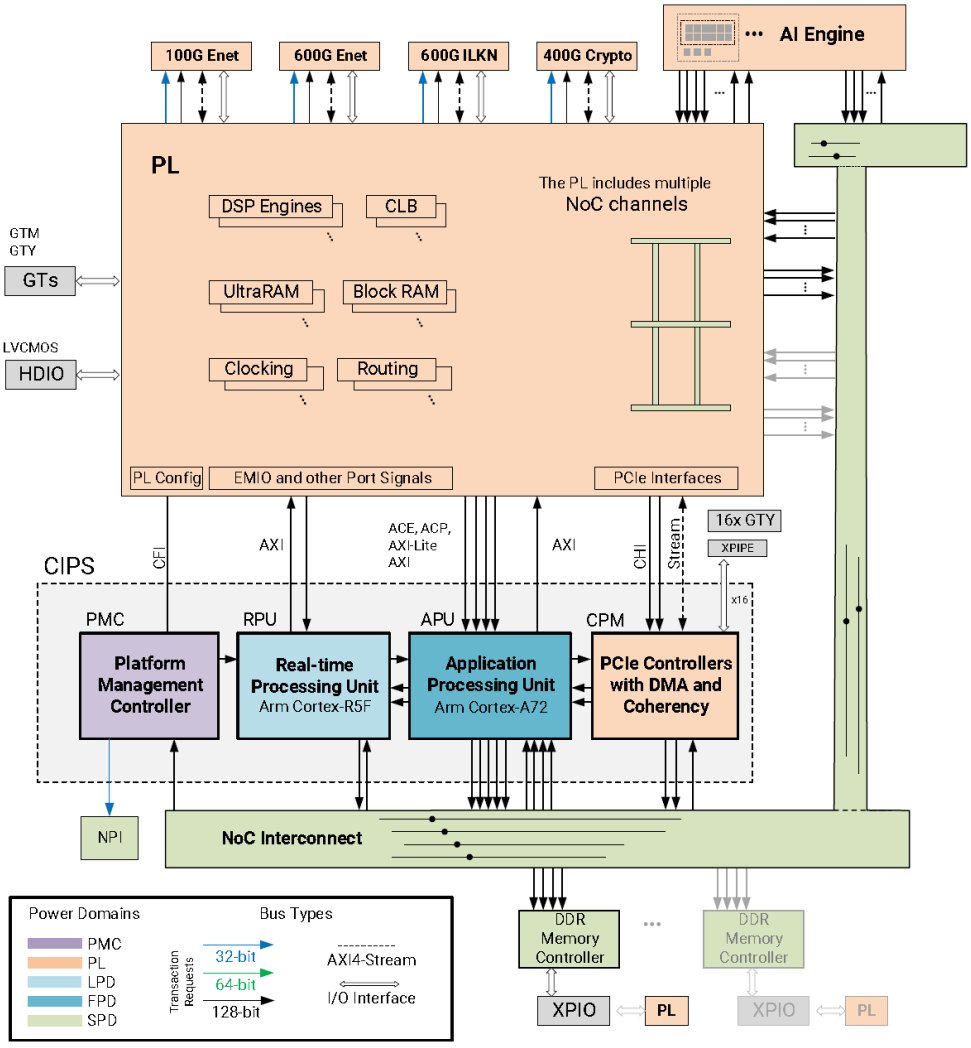

Figura 7.1: Connessioni con le altre componenti del dispositivo [4, Pag.80]

Configurable Logic Block

L'idea alla base della logica programmabile è molto semplice e si fonda sui blocchi logici configurabili. Ciascun blocco ha 32 tabelle di verità e 64 flip flop. Durante l'avvio del dispositivo è possibile configurare le tabelle di verità per realizzare una qualsiasi funzione di logica booleana. Specificando fino a sei input e due output si può descrivere l'output desiderato per ogni combinazione di input. I valori delle funzioni possono essere ricordati e trasferiti ad altri blocchi logici sfruttando i 64 flip flop.
Grazie ad un sistema di flip flop, riporti (carry) e multiplexer, l'output di una tabella può essere inserito negli input di altre tabelle, consentendo di estendere la complessità della funzione realizzata.
Il vantaggio delle tabelle di verità, chiamate look-up table (LUT), è che non richiedono alcun tipo di processore o elaborazione dei dati, quindi anche funzioni complesse vengono svolte negli stessi tempi di funzioni booleane base.
Durante la configurazione è possibile assegnare a 16 delle 32 look-up tables delle funzioni speciali. Ciascuna tabella può comportarsi come una memoria RAM da 64bit, un registro di shift a 32bit o due registri di shift a 16 bit. Queste 16 tabelle prendono il nome di LUTRAM o SLR-Capable LUT.

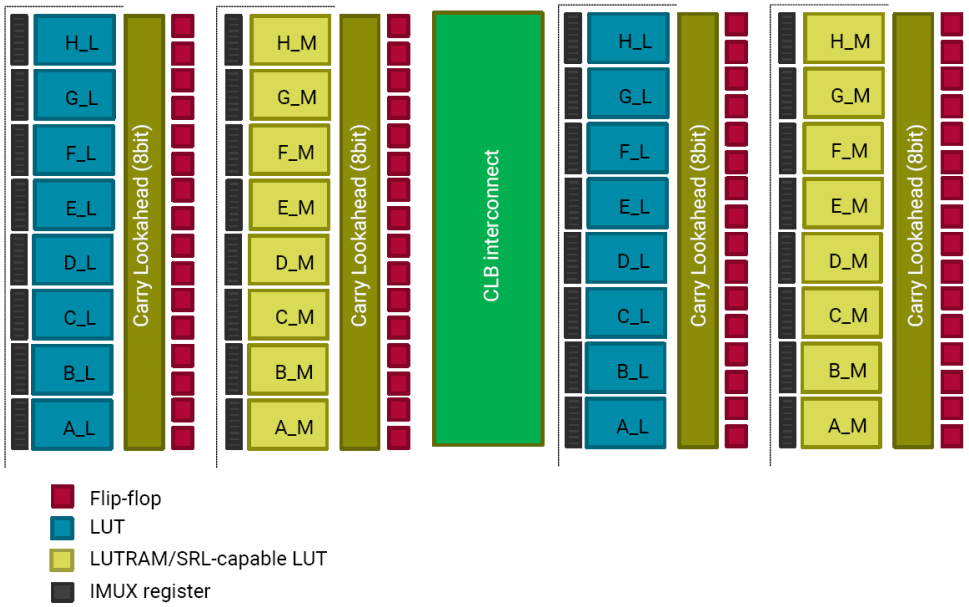

Figura 7.2: Schema di un singolo blocco configurabile [24, Pag.6]

Digital Signal Processing

Le applicazioni di elaborazione dei segnali digitali richiedono processori con moltiplicatori e accumulatori binari. L'utilizzo di CPU tradizionali risulta inefficiente in questo ambito. Per questo in tutti i dispositivi Versal la logica programmabile integra dei processori DSP.

I DSP sono realizzati con l'architettura DSP58 di Xilinx [25].

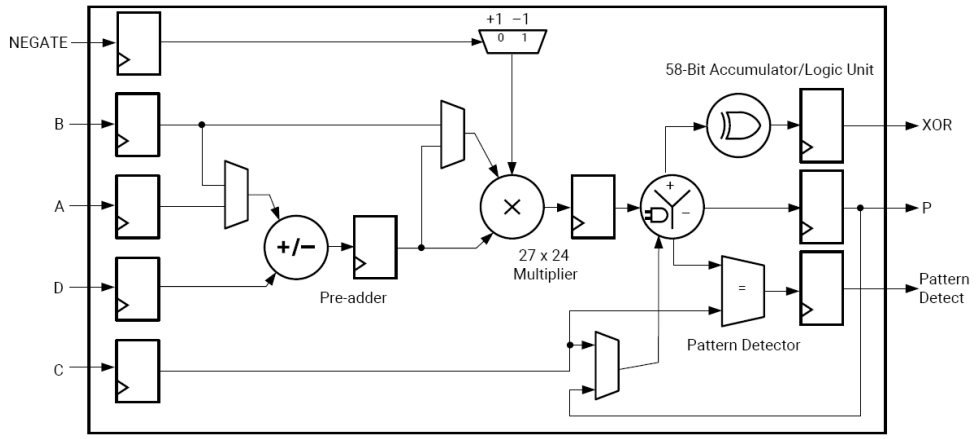

Figura 7.3: Architettura DSP58
[26, Pag.8]

L'architettura si basa su un moltiplicatore a 27 x 24 bit in complemento a due e un accumulatore a 58 bit. Integra anche un presommatore a 27 bit e la possibilità di invertire il segno del risultato tramite la porta negate. La grandezza dei due numeri in input (27 e 24 bit) è più che sufficiente per la maggior parte delle applicazioni di signal processing.
Ad arricchire le funzionalità vi sono un'unità logica e un pattern detector.
L'unità logica può eseguire tutte le funzioni logiche di base sui due input. Vede la sua utilità nelle implementazioni di Forward Error Correction e algoritmi di controllo tramite ridondanza ciclica.
Il pattern detector a 58 bit determina se il risultato presenta un determinato pattern. Può essere usato per il rilevamento di overflow o underflow e effettuare arrotondamenti convergenti e simmetrici.

L'impiego tipico di un DSP sono le operazioni con numeri in rappresentazione intera. L'architettura DSP58 sblocca anche la possibilità di effettuare operazioni con altri tipi di dati:

- Prodotto scalare tra tre interi a 8bit.
  Il moltiplicatore effettua il prodotto scalare considerando gli input come tre vettori.
- Operazioni su numeri complessi.
  Combinando due DSP58 adiacenti è possibile effettuare la moltiplicazione tra numeri complessi di 18 bit, con la possibilità di coniugare l'input.
- Operazioni su numeri in virgola mobile a singola precisione.
  Il moltiplicatore e il sommatore sono configurabili per fare operazioni in virgola mobile. Le moltiplicazioni possono avvenire su dati a 32 bit o a 16 bit.

RAM e Ultra RAM

Gli input delle tabelle di verità possono essere anche valori salvati in delle memorie RAM di appoggio. Nella logica programmabile sono presenti molti blocchi di RAM e Ultra RAM, disposti in colonne che attraversano la sezione.
Ciascun blocco di RAM è formato da due banchi da 18Kbit dotati di due porte di accesso. Le porte sono indipendenti quindi la lettura e la scrittura possono avvenire in modo asincrono correndo il rischio di sovrapporsi. Inoltre le porte hanno accesso a tutti i 36Kb del blocco RAM, diventa quindi compito della configurazione iniziale specificare il modo in cui vengono gestite le collisioni. Una scrittura può interrompere e sovrascrivere altre scritture precedentemente iniziate oppure essere ignorata. Se invece avviene una lettura durante una scrittura può essere annullata oppure effettuata sul vecchio valore vecchio in quanto viene aggiornato solo a scrittura completata.

Le memorie Ultra RAM invece non sono composte di banchi, ma sono memorie uniche di 228Kb. Sono delle memorie RAM a tutti gli effetti ma hanno una densità di memoria molto più elevata. Gli accessi avvengono in modo sincronizzato quindi non si pone il problema di collisioni.

Entrambe le memorie possono essere messe in cascata a formare memorie più grandi. Ad esempio in una colonna tipicamente sono presenti 24 blocchi di Ultra RAM, quindi volendo combinare i blocchi si ottiene una memoria di 5472Kbit. Le colonne poi possono essere di nuovo unite tramite la logica programmabile ottenendo discrete dimensioni di memoria RAM senza sacrificare velocità di accesso e prestazioni.

8. Comunicazione PCIe

I chip Versal sono pensati per essere installati su schede elettroniche simili alle schede madri dei computer. In alcune applicazioni i chip possono essere usati come System on Chip riuscendo a rispondere da soli alle necessità dell'impiego. In altri casi, come in server con elevato carico computazionale, i dispositivi Versal vengono integrati a bordo di schede da affiancare a CPU e GPU principali. La soluzione più semplice è sfruttare l'interfaccia PCIe in quanto è molto diffusa e consente un traffico di dati elevato. Per questo Xilinx ha inserito nei chip dei blocchi dediti alla comunicazione PCIe che garantiscono una comunicazione veloce e stabile con il sistema da accelerare.
L'implementazione dell'interfaccia può essere fatta da un blocco dedicato all'interno della logica programmabile, oppure da un blocco CPM. La presenza di questi blocchi dipende dal chip ed è consultabile in un datasheet [27, Pag.3]. In alcuni dispositivi entrambi i blocchi sono presenti, sarà l'utente a decidere quale utilizzare per la comunicazione.

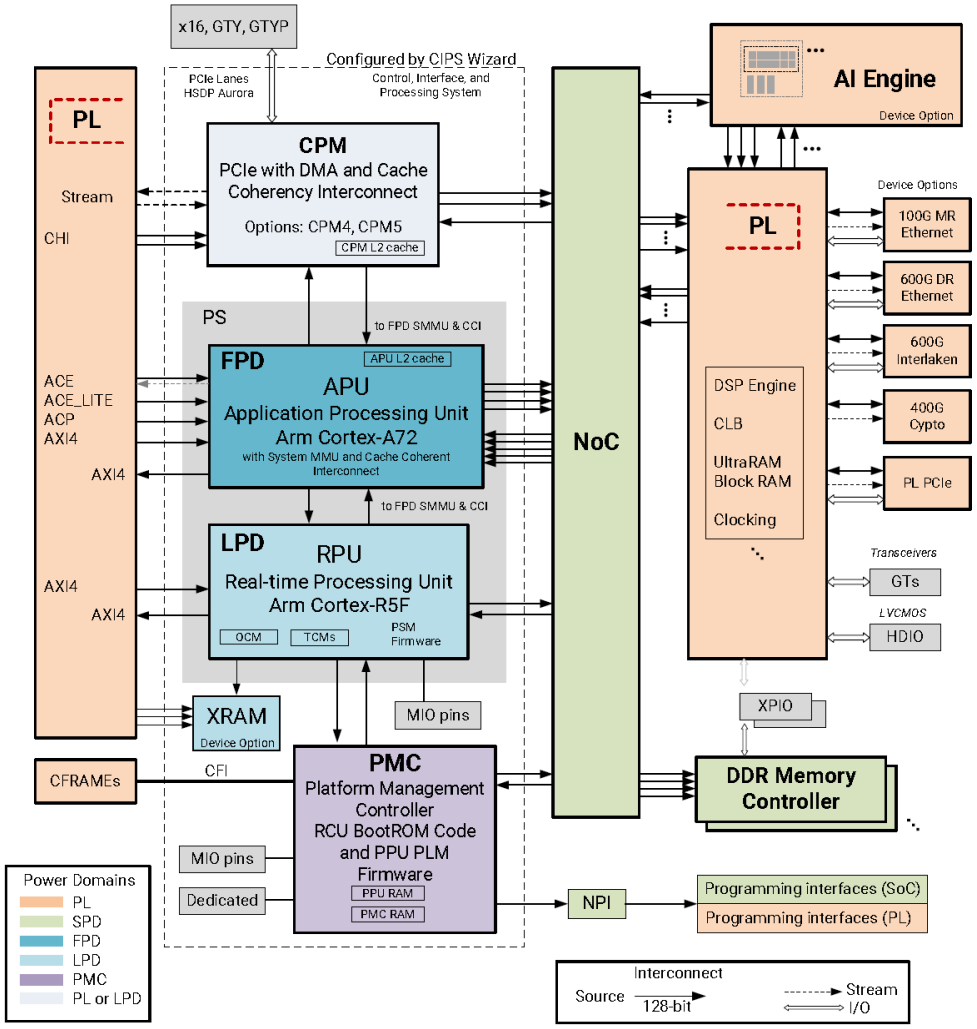

Figura 8.1: Architettura Versal
[4, Pag.46]

8.1 PL PCIe

I blocchi PL PCIe, realizzati su logica programmabile, supportano le due generazioni più recenti dell'interfaccia, ovvero PCIe 4.0 e 5.0. La differenza tra le due è la velocità di trasferimento dei dati, che sono 16GT/s e 32GT/s (16 e 32 miliardi di trasferimenti al secondo).
Seguendo la filosofia del dispositivo anche questi blocchi possono essere configurati. Essendo parte della logica programmabile all'avvio possono essere specificati diversi parametri in merito a funzionamento, rate di trasferimento e numero di linee PCIe.

Per il trasferimento dei dati si appoggiano a dei ricetrasmettitori GTM. Sono blocchi con la capacità di trasmettere e ricevere dati e fanno da ponte durante le comunicazioni seriali. Riescono a rispettare diversi protocolli di comunicazione ad alta velocità, come PCIe, Ethernet e Aurora IP, difatti sono condivisi dai PL PCIe con l'interfaccia di comunicazione Ethernet.
Hanno il compito di generare o ricevere i segnali elettrici necessari per la comunicazione dei dati. La dimensione dei pacchetti che vengono inviati può essere specificata e può anche essere specificata la velocità di comunicazione, o meglio, il clock di campionamento dei segnali, da cui poi deriva il bit-rate. I trasmettitori sono in grado di prendere una frequenza di riferimento e moltiplicarla fino a 160 grazie ad un anello ad aggancio di fare. Riescono cosi a trasmettere fino a 112 Gb al secondo nel caso dei trasmettitori GTM o 32Gb/s per i GTY e GTYP.

Una volta raggiunta la logica programmabile i dati in ingresso hanno accesso al NoC e al resto del dispositivo. I blocchi di comunicazione sono collegati alla logica programmabile che introduce una leggera latenza nei trasferimenti. In alcune applicazioni è tollerabile, consentendo di rimuovere il costo del CPM, in altre è necessario avere una comunicazione PCIe diretta con processori e NoC.

8.2 CPM

In alcuni dispositivi Versal oltre ai blocchi PL PCIe viene reso disponibile il CPM. Si tratta di un blocco che permette di effettuare collegamenti a bassa latenza e banda elevata tramite l'interfaccia PCIe.

Analogamente a quanto visto per i blocchi della logica programmabile, il blocco CPM può implementare le due generazioni dell'interfaccia PCIe. Quarta e quinta generazione dell'interfaccia vengono implementate rispettivamente da CPM4 e CPM5.
Per garantire prestazioni elevate il CPM si compone di due controller PCIe con DMA. Entrambi sono configurabili in quanto a numero di linee di comunicazione ma l'interfaccia PCIe mette a disposizione solo 16 linee, quindi i due controller devono dividersele. Se un solo controller è abilitato ha a disposizione l'intera banda offerta dall'interfaccia a 16 linee.
Anche il CPM si appoggia su dei ricetrasmettitori, in particolare GTY per CPM4 e GTYP per CPM5. Il loro funzionamento è identico ai GTM, presentano però caratteristiche di bit-rate e latenza differenti. I GTY e GTYP sono riescono a trasferire meno velocemente i dati, però lo fanno in modo più efficiente e diretto con i processori, consentendo di ridurre la latenza. La minore velocità inoltre viene compensata dal fatto che sono presenti più ricetrasmettitori a disposizione solo del CPM, senza essere messi in comune ad altri protocolli come nel caso della logica programmabile. Essendo presenti più ricetrasmettitori anche il CPM è in grado di sfruttare a pieno la banda offerta dall'interfaccia PCIe.

Il grande vantaggio dei CPM è la presenza di un DMAC e un modulo CCIX. Questo consente ai dati di avere un accesso diretto alle memorie cache dei processori, mantenendo la coerenza dei dati e la facilità di trasferimento di un sistema DMA. I dati possono anche accedere direttamente al NoC per raggiungere l'AI Engine o la memoria RAM della scheda su cui monta il chip.

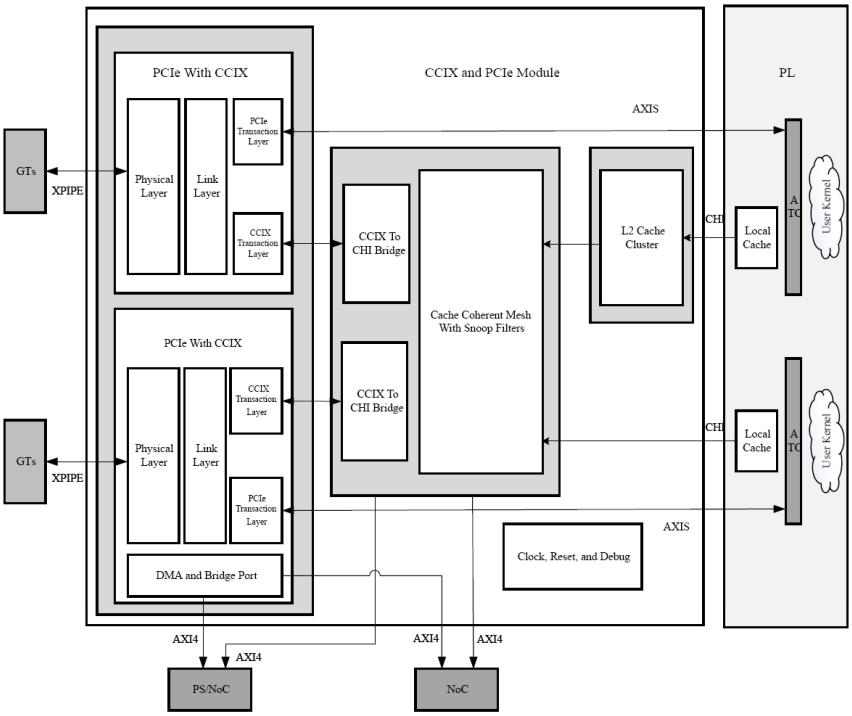

Figura 8.2: Schema del CPM4
[28, Pag.8]

Inoltre, a differenza dell'implementazione su logica programmabile, il CPM viene configurato separatamente, rendendo quindi il blocco operativo in modo rapido subito dopo il boot, senza necessità di attendere la configurazione della PL.

La presenza dei blocchi CCIX servono a implementare la comunicazione secondo il protocollo CCIX [29].
Cercando di stabilire una connessione tra due dispositivi (Versal e dispositivo da accelerare) si pone il problema di creare un modo efficiente per dialogare, infatti la complessità dei sistemi ibridi si rispecchia anche nel software da sviluppare per utilizzarli. Sulla base di questa necessità è stata creata l'interfaccia CCIX che consente di ottimizzare e semplificare come vengono definiti i sistemi eterogenei. Il CCIX è un protocollo applicabile anche su PCIe ed è essenziale per programmare efficientemente i dispositivi Versal. Questo si traduce anche nella possibilità di mettere in comunicazione dispositivi con set di istruzioni diversi.

I CPM4 implementano la prima generazione dell'interfaccia mentre i CPM5 sfruttano la CCIX 1.1, anche chiamata CLX. Le differenze tra CCIX e CLX sono minime, aumenta leggermente la banda disponibile e la facilità di comunicazione.
Grazie a controller dedicati, il DMA e il protocollo CCIX diviene vantaggioso usare il CPM come interfaccia PCIe primaria per i dispositivi Versal che lo rendono possibile.

9. Network on Chip

Il NoC programmabile è un network di interconnect (punti di accesso) basato sull'interfaccia AXI4 che consente un dialogo ad alta velocità tra tutte le parti del dispositivo. E' stato progettato per essere la principale via di comunicazione tra le varie sezioni del dispositivo.
Senza il NoC i vari processori visti finora, ovvero processing system, logica programmabile e AI Engine non potrebbero comunicare efficientemente tra di loro perché la condivisione dei dati presenti nelle varie memorie sarebbe molto più complessa e lenta.

Alcune delle connessioni realizzate dal NoC ad esempio sono:

- Condividere l'uso della memoria DRAM globale del dispositivo
- Connessioni tra varie zone della logica programmabile
- Accesso preciso ai vari moduli di memoria dell'AI Engine Array
- Consentire al PS di accedere alla logica programmabile e alla DRAM

Il network si compone di NoC orizzontale[1], distribuito sulla cima e sul fondo del dispositivo e NoC verticale[2], composto da un numero variabile di colonne, fino ad 8, a seconda del dispositivo.
Il NoC deve essere configurato e programmato tramite l'interfaccia di programmazione NPI[3] durante l'avvio del dispositivo. La configurazione viene effettuata in automatico dal PLM, leggendo gli appositi file NPI presenti nel dispositivo di boot. Possono essere programmati i registri del network che definiscono le tabelle di routing, clock e impostazioni del QoS.

Ogni connessione che attraversa il NoC ha un Quality Of Service associato. Si tratta di un requisito che il NoC cerca di garantire per le varie connessioni. Durante la configurazione i vari valori di QoS delle connessioni sono utilizzati per ottimizzare l'utilizzo delle risorse del NoC.

Lo scopo principale del NoC è di far arrivare ai processori i dati presenti nella memoria RAM della scheda su cui monta il chip. Nonostante siano presenti molte memorie all'interno del chip, la loro dimensione non è minimamente vicina a quella necessaria per la scrittura di un intero programma. Sono infatti memorie RAM altamente veloci ma con basse densità di memoria. Solo piccole parti di programma possono essere scritte e spesso vengono utilizzate solo come memorie di appoggio dei dati. La vera memoria che contiene il programma da eseguire è collocata sulla scheda in cui è installato il chip.
Al fine di interfacciarsi con la RAM principale esterna sono presenti fino a 4 controller per memore RAM DDR[4]. Attraverso i DDRMC i vari processori, detti master, possono accedere alla RAM principale.

L'accesso al NoC avviene tramite due unità, le NoC master unit (NMU) e le NoC slave unit (NSU). Esse rappresentano le porte di ingresso e uscita per le varie sezioni del dispositivo. Le unità si occupano di elaborare i pacchetti AXI-4 e AXI-Stream provenienti dai processori inserendo delle informazioni essenziali per il viaggio all'interno del NoC. L'elaborazione consiste nell'aggiunta di 24 bit che specificano unità sorgente e unità di destinazione. Durante l'elaborazione viene anche controllata l'integrità della transazione e dei dati trasmessi.

[1] HNoC
[2] VNoC
[3] NoC Programming Interface
[4] DDRMC

I percorsi del NoC raggiungono molte zone del dispositivo. Se i punti di accesso fossero pochi sarebbe possibile creare collegamenti diretti e indipendenti tra ogni master e i vari slave.
Data la complessità del chip Versal e l'elevato numero di porte d'accesso al NoC, adottare questa soluzione è inefficiente e complicata. Per questo il sistema usato nel NoC è simile a quello di un vero e proprio network informatico. I vari punti di accesso e di uscita dal NoC sono collegati direttamente a degli switch, ovvero blocchi funzionali chiamati NPS, il cui funzionamento è analogo agli switch utilizzati nelle reti internet. Il loro compito quindi è indirizzare i pacchetti alla destinazione corretta senza creare collisioni con altri dati che viaggiano nel NoC.
Per instradare correttamente i dati, gli switch sfruttano un sistema di indirizzi a 12 bit per sorgente e destinazione. Confrontando gli indirizzi con delle tabelle di routing gli switch sapranno qual'è la strada migliore da far percorrere ai dati.

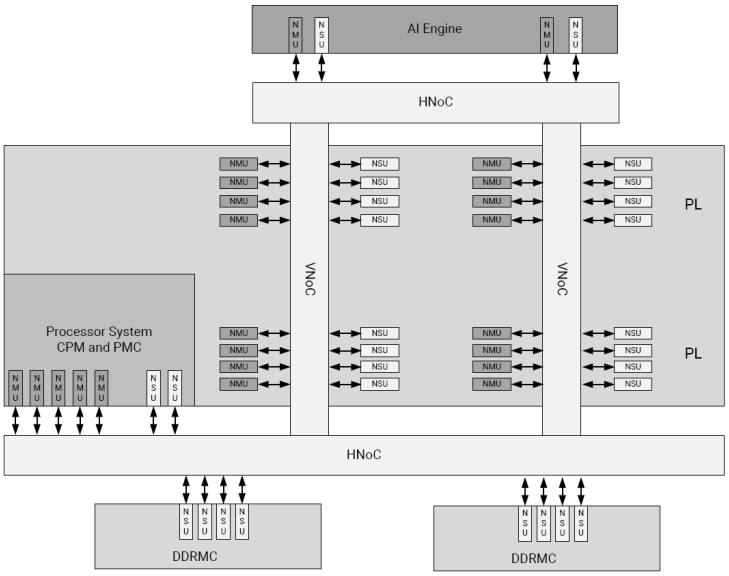

Figura 9.1: Master e slave delle varie sezioni [30, Pag.7]

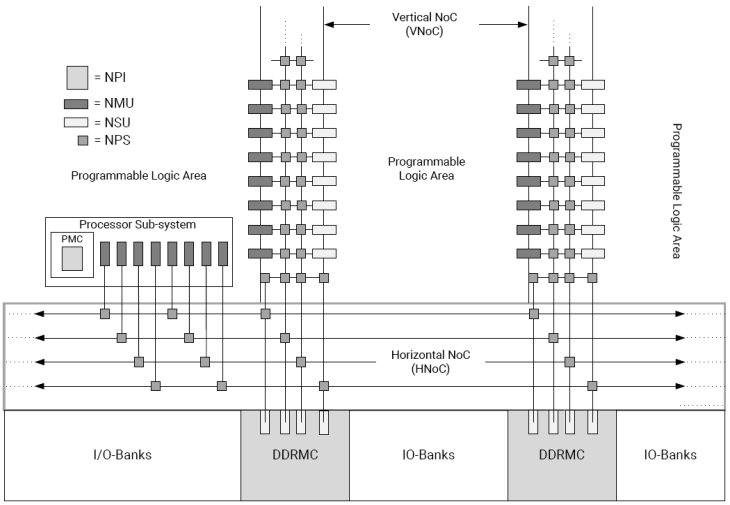

Figura 9.2: Percorsi del NoC [30, Pag.14]

9.1 DDR Memory Controller

Il NoC si interfaccia con la memoria RAM tramite uno o più DDRMC. Si tratta di controllore ad alta efficienza e bassa latenza che può essere ottimizzato a seconda delle applicazioni da eseguire.
I DDRMC sono una risorsa condivisa da tutto il dispositivo tramite il NoC. Durante l'esecuzione di un programma le istruzioni e le variabili da elaborare vengono salvate sia sulla cache che sulla memoria RAM esterna. Il passaggio di dati tra i processori avviene tramite il CCI oppure tramite la memoria RAM quando i dati non sono nelle cache.

Il controller è stato progettato per memorie RAM comuni come DDR4, LPDDR4 e LPDDR4X a 4266Mb/s, ovvero formati e frequenze che si possono trovare in qualsiasi dispositivo come computer, tablet e laptop.
A seconda del dispositivo possono essere presenti fino a quattro DDR memory controller. Il funzionamento con più di un controller si basa sulla suddivisione dei compiti. Le richieste effettuate dalle unità del NoC vengono divise in 2,3 o 4 blocchi e ogni DDRMC poi prenderà in carico un blocco. La suddivisione del carico di lavoro consente di effettuare trasferimenti fino a 4 volte più veloci.
Anche se son presenti più controller i vari processori non devono preoccuparsi di dividere i pacchetti. I vari DDRMCs vengono visti come un singolo punto di accesso ed è compito del NoC comunicare con i DDRMC per dividere al meglio le richieste.
Questo significa che anche per il DDRMC è possibile impostare i parametri di QoS. Esiste infatti un sistema di suddivisione e gestione prioritaria delle richieste di accesso che si basa proprio sui valori di QoS.
La configurazione ottimale delle interfacce di memoria, quindi lunghezza dei pacchetti, tipo e frequenza di lettura, avviene in fase di avvio e può essere identificata grazie ad un software di Xilinx.

Una peculiarità del DDRMC è la presenza di una unità di protezione per la memoria di Xilinx, chiamata XMPU. Serve a garantire che solo le transazioni con ID e formato corretti possano essere eseguite, per evitare di lasciare transizioni in sospeso o che richiedano troppo tempo prima di venire annullate. Impedisce anche che vadano a finire nella destinazione sbagliata proteggendo le zone di memoria.

Conclusioni

La versatilità dei dispositivi in combinazione alla facilità di utilizzo e implementazione rende la famiglia di dispositivi Versal una soluzione che si applica a molti ambiti del mercato. Se sfruttati a pieno riescono a offrire accelerazioni consistenti in svariate applicazioni, consentendo a qualsiasi sistema di essere al passo con gli standard di velocità e prestazioni.

I tre tipi di processori integrati nei chip Versal, ovvero processing system, logica programmabile e AI Engine, offrono differenti capacità computazionali per coprire una vasta gamma di settori. Il processing system è tipicamente usato per applicazioni di controllo, sistemi operativi e interfacce di comunicazione. La logica programmabile si occupa di elaborazione e trasferimento dati, funzioni personalizzate e processi non basati su vettori. L'AI Engine invece eccelle in applicazioni vettoriali complesse come elaborazione dei segnali digitali, intelligenza artificiale, machine learning e comunicazione 5G. La loro combinazione forma un sistema computazionale completo e ricco di opportunità d'impiego.

Al di fuori delle applicazioni vettoriali l'AI Engine non è efficiente, per questo viene affiancato dalla logica programmabile. Le due sezioni sono pensate per lavorare insieme appropriandosi delle funzioni che incontrano i loro punti di forza. La loro unione consente ai dispositivi Versal di offrire versatilità nel tipo di carico di lavoro mantenendo performance e bande elevate.
I dispositivi che non dispongono dell'AI Engine rimangono una valida soluzione per ambiti in cui non sono richiesti processi vettoriali.

Le tecniche di machine learning stanno conquistando sempre più ambiti applicativi, le aziende stanno ricorrendo sempre più all'intelligenza artificiale. Ad esempio un'area in cui l'impiego del machine learning sta vedendo una tremenda crescita è lo streaming e l'elaborazione video. I servizi che offrono la possibilità di vedere film e serie TV online stanno diventando sempre più popolari negli ultimi anni. La grande scelta che i sempre più numerosi utenti hanno ha richiesto lo sviluppo di nuove tecniche per analizzare quali contenuti sono di moda, quali offrire ad un determinato tipo di pubblico in base alle preferenze di ogni singolo utente. L'elaborazione di questa grande quantità di dati richiede molto tempo se eseguita su una normale CPU. L'intelligenza artificiale invece si adatta perfettamente a questo tipo di compiti, in cui vi sono molti dati ed è necessario prendere decisioni adattate alle varie situazioni. L'elaborazione video invece può sfruttare l'intelligenza artificiale per il rendering o l'interpolazione dei fotogrammi.

Un altro ambito a cui si può pensare sono i social network. Per le aziende è fondamentale analizzare l'attività di migliaia di utenti per fornire contenuti personalizzati e migliorare l'esperienza d'utilizzo. Un altro esempio ancora sono gli assistenti vocali che stanno migliorando continuamente imparando dalle richieste degli utenti. L'intelligenza artificiale sembra quindi essere la soluzione di maggior interesse per il mercato del futuro, per questo in aziende in cui si mira ad offrire servizi di alto livello è importante ricorrere a processori come l'AI Engine.

Bibliografia

[1] *CISC*. URL: https://it.wikipedia.org/wiki/Complex_instruction_set_computer.

[2] *RISC*. URL: https://it.wikipedia.org/wiki/Reduced_instruction_set_computer.

[3] *White Paper: Versal ACAPs*. URL: https://www.xilinx.com/support/documentation/white%5C_papers/wp505-versal-acap.pdf.

[4] *Versal ACAP Technical Reference Manual*. URL: https://www.xilinx.com/support/documentation/architecture-manuals/am011-versal-acap-trm.pdf.

[5] *Versal ACAP System Software Developers Guide*. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug1304-versal-acap-ssdg.pdf.

[6] *eMMC*. URL: https://it.wikipedia.org/wiki/MultiMediaCard.

[7] *Interfaccia SPI*. URL: https://it.wikipedia.org/wiki/Serial_Peripheral_Interface.

[8] *MicroBlaze*. URL: https://www.xilinx.com/products/design-tools/microblaze.html.

[9] *Triple Modular Redundancy - TMR*. URL: https://www.xilinx.com/content/dam/xilinx/support/documentation/ip_documentation/tmr/v1_0/pg268-tmr.pdf.

[10] *ARM CoreLink GIC-500*. URL: https://developer.arm.com/documentation/ddi0516/e/introduction/about-the-gic-500.

[11] *Unità di gestione della memoria*. URL: https://it.wikipedia.org/wiki/Unit%C3%A0_di_gestione_della_memoria.

[12] *Cache Coherent Interconnect*. URL: https://developer.arm.com/ip-products/system-ip/corelink-interconnect/corelink-cache-coherent-interconnect-family/corelink-cci-500.

[13] *Cache Coherency*. URL: https://en.wikipedia.org/wiki/Cache_coherence.

[14] *AMBA AXI and ACE Protocol*. URL: https://developer.arm.com/documentation/ihi0022/hc.

[15] *Advanced Encryption Standard*.

[16] *Secure Hash Algoritm*. URL: https://it.wikipedia.org/wiki/Secure_Hash_Algorithm.

[17] *Tightly-coupled memory*. URL: https://developer.arm.com/documentation/ddi0338/g/level-one-memory-system/tightly-coupled-memory.

[18] *Soft Error*. URL: https://en.wikipedia.org/wiki/Soft_error.

[19] *ARM GIC-390*. URL: https://documentation-service.arm.com/static/5e8e1d3588295d1e18d3635c?token=.

[20] *Versal ACAP Ai Engine*. URL: https://www.xilinx.com/support/documentation/architecture-manuals/am009-versal-ai-engine.pdf.

[21] *PLL - Anello ad aggancio di fase*. URL: https://it.wikipedia.org/wiki/Phase-locked_loop.

[22] *AXI4 Stream Protocol*. URL: https://developer.arm.com/documentation/ihi0051/a/Introduction/About-the-AXI4-Stream-protocol.

[23] *Sincronizzazione lock*. URL: https://it.wikipedia.org/wiki/Lock.

[24] *Versal ACAP Configurable Logic Block*. URL: https://www.xilinx.com/support/documentation/architecture-manuals/am005-versal-clb.pdf.

[25] *DSP Engine*. URL: https://www.xilinx.com/support/documentation/architecture-manuals/am004-versal-dsp-engine.pdf.

[26] *Versal ACAP DSP Engine*. URL: https://www.xilinx.com/support/documentation/architecture-manuals/am004-versal-dsp-engine.pdf.

[27] *Versal datasheet*. URL: https://www.xilinx.com/support/documentation/data_sheets/ds950-versal-overview.pdf.

[28] *Versal ACAP CPM CCIX*. URL: https://www.xilinx.com/support/documentation/architecture-manuals/am016-versal-cpm-ccix.pdf.

[29] *CCIX*. URL: https://www.ccixconsortium.com/wp-content/uploads/2019/11/CCIX-White-Paper-Rev111219.pdf.

[30] *Versal ACAP Programmable Network on Chip and Integrated Memory Controller v1.0*. URL: https://www.xilinx.com/support/documentation/ip_documentation/axi_noc/v1_0/pg313-network-on-chip.pdf.